ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN

Titulación:

INGENIERO TÉCNICO EN INFORMÁTICA DE GESTIÓN

Título del proyecto:

SISTEMA DE ILUMINACIÓN VIARIA ADAPTATIVA

Iñaki Etxeberria Ruesta

Tutor: José Javier Astrain Escola

Pamplona, 17 de julio de 2013

ÍNDICE

INTRODUCCIÓN

Antecedentes

Desde hace ya años se trata de reducir la necesidad de consumo energético. Por varias razones: ecología, ahorro y en el presente por la actual situación de crisis económica. Estas razones no son decisivas en el desarrollo de este proyecto y nos basaremos en la premisa del "desarrollo sostenible".

Este proyecto aborda la tarea de la iluminación en vías públicas desde un punto de vista de la urbótica. Desarrollar e implementar un sistema que cumpla con las necesidades de iluminación de un tramo de vía a un coste mínimo. Por coste entendemos el coste total de implementación así como costes de operación y suministro eléctrico. Y por necesidades de iluminación entendemos la necesidad inherente del ser humano de ver la calle por la que se desplaza.

Un sistema cuyo coste total nunca se llegue a amortizar o peor aún, que haga que el consumo se incremente no es interesante.

Asimismo un sistema que no ilumina la vía suficientemente o que lo hace cuando hay suficiente luz natural no es adecuado tampoco.

En una sola frase: el sistema se tendrá que ocupar de que la vía esté iluminada con una cantidad de luz adecuada a las circunstancias y con un consumo mínimo.

El consumo de energía eléctrica con el propósito de iluminar los lugares públicos se ha venido reduciendo principalmente por el avance tecnológico en los elementos lumínicos, en su precio, consumo unitario, etc. Pero el modo de iluminación no ha variado mucho. Llegada una hora concreta se encendían las luminarias y estas permanecían encendidas hasta la hora de apagado.

Después vinieron los sensores de luz, que activaban las farolas por debajo de un umbral de luminosidad prefijado. Y hasta que dicho umbral no se rebasaba, el sistema permanecía encendido constantemente y a la misma potencia (y consumo).

En los últimos años se han venido instalando en algunas ciudades sistemas de iluminación es que estos se adecuen a las necesidades puntuales de la vía a iluminar. Según el tránsito de personas en concreto.

Objetivos y alcance

El objetivo de este proyecto es desarrollar un sistema que monitorice el estado de tránsito e iluminación de un tramo de vía y según los parámetros disponibles actuar sobre la iluminación de dicho tramo.

Otro objetivo es también el de reducir el consumo eléctrico y alargar la vida útil de los elementos lumínicos, al no estar constantemente sometidos a la misma potencia de funcionamiento. Optimizar los recursos para ofrecer el mismo servicio con un coste mínimo es la tendencia a seguir.

Además el sistema funcionará de forma autónoma, sin que sea necesaria la intervención externa para su funcionamiento normal. La red de sensores decidirá en cada momento si iluminar la vía y con qué cantidad de luz debe hacerlo.

Este proyecto no puede tomarse como una solución global a la tarea de iluminación viaria. Según la Guía Técnica de Eficiencia Energética en Iluminación, previsto en el RD 1890/2008, hay una gran variedad de factores a la hora de abordar la iluminación pública, como son el tipo de vía, luminarias a utilizar, volumen del tráfico, características de la vía etc. Y que se deberán tener en cuenta para adecuar el sistema.

ESTADO DEL ARTE

Hoy en día la mayoría de la iluminación de nuestras calles no dispone de un sistema inteligente que las regule, de manera que llegado el ocaso, esta se enciende y se mantiene a una potencia constante toda la noche.

Son varias las soluciones que se han adoptado durante los últimos años con el fin de conseguir el ansiado ahorro energético:

En las Islas Canarias por ejemplo, las luminarias disponen de dos sistemas de iluminación: uno normal, que consiste en lámparas de vapor de sodio de alta presión y otro de baja presión que a media noche se activa. Pese a que los elementos lumínicos son más caros, por disponer de ambos sistemas en un mismo elemento, el ahorro en la factura eléctrica es considerable y pronto se amortiza la inversión. Sin embargo, presenta un gran defecto y es que con el sistema de bajo consumo la percepción de los colores se ve disminuida al tratarse de luz monocromática.
Aun así, durante años se ha mantenido este sistema, pues se prima la baja contaminación lumínica en pro del observatorio astronómico.

Otra solución es la de instalar un doble circuito en las luminarias y llegada una hora, apagar uno de los dos circuitos, quedando así la mitad de las lámparas encendidas. Aunque efectivo por la reducción directa en el consumo eléctrico, se desaconseja su uso. Primero por el sobrecoste que supone instalar un doble circuito y porque se producen zonas oscuras que en carretera se consideran muy peligrosas.

Interruptor crepuscular. Este es el sistema más extendido. Un sensor de luz actúa a modo de interruptor, encendiendo las farolas cuando la luz natural baja de un umbral preestablecido. Cuando dicho umbra se rebasa, el sistema apagará las luminarias.

En los últimos años se ha venido sustituyendo las tradicionales bombillas por otras de tecnología LED. No obstante la sustitución no puede ser indiscriminada y tiene que acompañarse de un estudio específico que ya se ha determinado en el documento “Requerimientos técnicos exigibles para luminarias con tecnología LED de alumbrado exterior” del Comité Español de la iluminación. En resumidas cuentas se pone de manifiesto que la fotometría de una luminaria LED no es la misma que la de una de luminaria única y que esta se debe traducir.

PROPUESTA

La decisión de instalar iluminación exterior no es trivial, existen múltiples factores a tener en cuenta. En general se puede decir que se persigue suplir la falta de iluminación natural y hacerlo de la manera más eficiente. Como ya he mencionado, este proyecto no contempla aspectos luminotécnicos, por lo que podría ser complementario a un proyecto de instalación lumínica o de mejora de una ya existente.

Un aspecto que está en auge es la monitorización de las vías públicas y por ello este proyecto propone adecuar la iluminación viaria según las necesidades en cada momento. En mi opinión dos aspectos básicos a tener en cuenta para adecuar la iluminación de un tramo de calle son: la luminosidad natural y por otro el tráfico existente. Ambas son variables en el tiempo, el tráfico además es impredecible, por lo que la monitorización debe ser constante. Se debe considerar asimismo que hay ciertas horas del día en que la iluminación natural es suficiente, por lo que en esas franjas horarias no es necesario que el sistema monitorice la vía.

Así pues me propongo diseñar un sistema que con al menos esos dos parámetros sea capar de regular la cantidad de luz emitida. Veamos aspectos concretos de mi propuesta:

Interruptor crepuscular: similar al modo de funcionamiento actual, cuando la cantidad de luz baje del umbral establecido se encenderá el sistema, que no las luminarias. Durante el día, el sistema puede permanecer en reposo, ahorrando a su vez energía. Cuando la luz comience a decrecer, el sistema se encenderá y comenzará a monitorizar la calle.

Trafico: rodado o a pie. Cuando hay movimiento en la calle, el sistema ha de proporcionar más luz y sin embargo si no hay nadie, bastará con una luz mínima. Este factor solo se tendrá en cuenta cuando el sistema esté encendido, es decir en las horas en las que la iluminación no sea suficiente.

Entre la luz mínima o reposo y el máximo de iluminación que pueden proporcionar las luminarias, existe un amplio rango de potencia a suministrar que se determinará por ambos factores. Cuanta menos luz crepuscular haya, mayor será la iluminación artificial y a su vez cuanto mayor tráfico exista, más iluminada deberá estar la calle.

El sistema ha de contar con elementos que sean capaces de obtener los dos parámetros de luz natural y tráfico. Será suficiente con tener un sensor de luz por zona pero será necesario más de un sensor de presencia para adecuar la luminosidad en cada tramo de la vía. Si es muy larga, que una persona este andando por un extremo de la vía no exige que el otro extremo se ilumine.

Debemos diseñar pues una red de sensores que sean capaces de comunicarse entre sí. Para ello y como requisito funcional tenemos unas placas programables de la empresa Libelium: Waspmote. A estas placas se acoplaran sensores de presencia, un sensor de luminosidad, antenas para la comunicación entre placas y un regulador de potencia para la luminaria. La comunicación entre las placas es necesaria para trasmitir información de la luminosidad en cada momento y para comunicar a los vecinos la presencia de personas o vehículos y que estos iluminen la zona delante de ellos.

Se desea un sistema descentralizado, de manera que cada placa sea capaz de actuar sobre su luminaria asociada de forma independiente.

ANALISIS Y ESPECIFICACION DE REQUISITOS

Metodología

Para este proyecto se ha elegido el método de desarrollo de programación extrema. La idea es disponer de un prototipo del sistema rápido y simple y que permita ver su funcionamiento básico. A partir de un rápido análisis, ir avanzando poco a poco en el desarrollo e ir adaptando el proyecto a medida que este crezca.

En esta metodología se busca la simpleza en todo el conjunto: un diseño simple y un código simple que permitan ir creciendo sin tener que hacer demasiados cambios. Un diseño complejo junto con sucesivos cambios aumenta exponencialmente la complejidad de todo el proyecto. Así el código se comentará lo mínimo y en la medida de lo posible será autodocumentado o lo que es lo mismo: utilizar nombres para las variables y funciones que den a entender su funcionalidad. Utilizar nombres largos no decrementa la eficiencia del código pero sí que facilitan la comprensión del mismo.

Además un contacto continuo con mi tutor del proyecto me ha permitirá aumentar progresivamente el proyecto de forma controlada, sin desviarme de los objetivos. Las reuniones no serán programadas, sino que se realizarán cuando se alcance un hito. Tras revisar el avance, se validará y se procederá a continuar con la siguiente tarea.

Lo primero que hemos de conseguir es que las placas puedan comunicarse entre sí, pues se sabe que deberán transmitir información entre ellas. Después recopilar información acerca de la luminosidad ambiental y el tráfico del lugar. Y por último que esa información se comparta entre todas las placas.

Requisitos funcionales

- El sistema ha de funcionar de manera autónoma en condiciones normales.
- Debe ser capaz de iluminar de manera adecuada la vía en cada momento, y reducir el consumo de electricidad destinado a iluminarla.
- La potencia con la que iluminen las luminarias vendrá determinada por los datos recopilados por el sistema. Los básicos son: luz natural y presencia de personas o vehículos.
- Las placas funcionaran de manera independiente pero coordinada con el resto de placas.

Requisitos no funcionales

- El sistema ha de ser mantenido fácilmente. Está ideado para ser instalado en farolas, en lo alto de estas previsiblemente, por lo que el acceso es algo complicado. Se tratará de usar algún tipo de mantenimiento sin cables.
- El sistema ha de ser económico o que se rentabilice a corto-medio plazo. Junto con el requisito anterior, ha de ser barato de mantener.
- El código fuente ha de ser lo más simple posible y fácil de entender, con el objetivo de adecuar lo más fácilmente posible el sistema a cada situación.

Requisitos adicionales

- El sistema ha de desarrollarse con placas Waspmote y la comunicación entre ellas se hará con tecnología ZIGBEE. Para el prototipo es un requisito impuesto, aunque la tecnología de comunicación se puede variar fácilmente.

DISEÑO E IMPLEMENTACIÓN

Fase 0: funcionamiento del sistema e introducción a los dispositivos

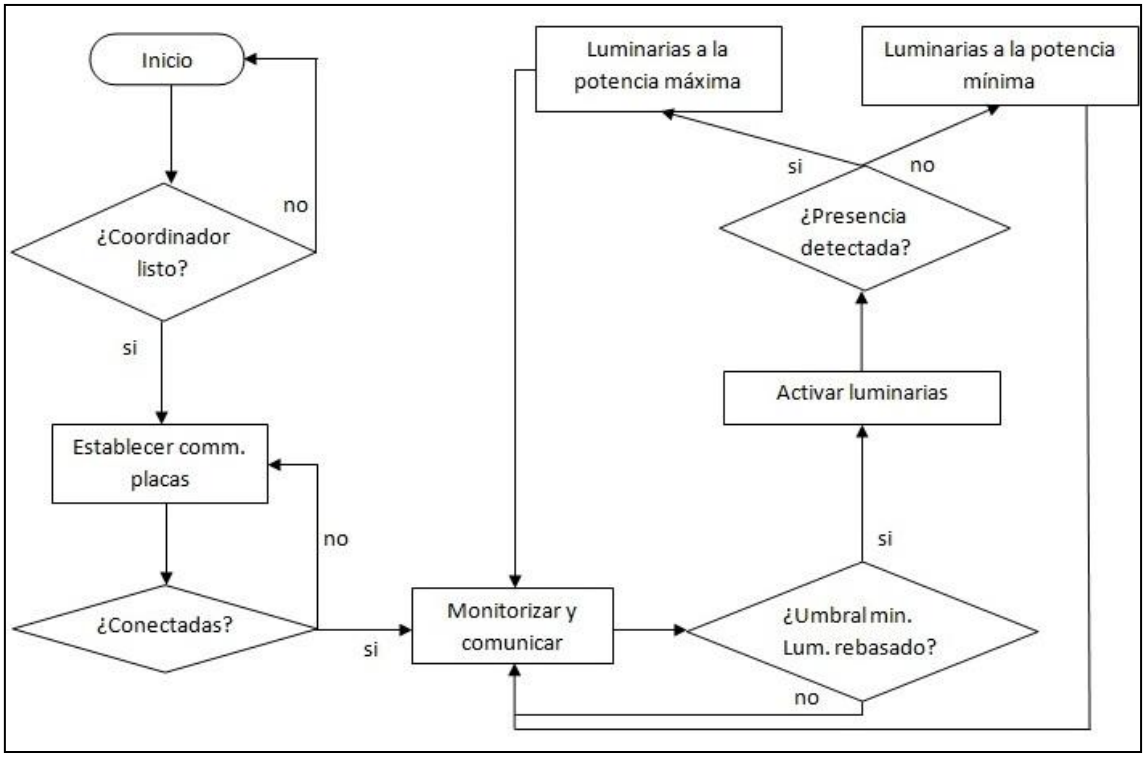


Figura 0: diagrama de flujo. Funcionamiento del sistema.

En la figura 0 podemos comprobar el funcionamiento del sistema, mediante un diagrama de flujo. Es un funcionamiento simple pero efectivo. El sistema sólo actuará cuando la luminancia baje de un nivel fijado. Así se conseguirá: ahorrar en el consumo del propio sistema; ahorrar en el consumo eléctrico de las luminarias, pues se encenderán cuando sea necesario, no cuando llegue la hora; ahorrar aún más iluminando sólo aquellas luminarias que sean necesarias, en el instante necesario, manteniendo una iluminación mínima cuando por la vía no circule nadie.

El sistema se montará con placas de la empresa Libelium. Radicada en Zaragoza, ofrece soluciones OpenSource para todo tipo de proyectos. Están orientados a lo que se denomina M2M (Machine to Machine) o intercambio de información entre dos máquinas remotas. En un concepto bastante genérico pero que tiene su relevancia en estos proyectos. La mayoría suelen constar de una red de sensores que funcionan de forma autónoma y deben comunicar los datos que recopilan.

Comercializan tres dispositivos diferentes:

- Meshlium, ver figura 1

Figura 1: Meshlium

Su función principal es la de ser el punto central de una red de sensores, a través de enlaces wireless. Dispone de antenas para Zigbee, WIFI, Bluetooth, 3G/GPRS y un módulo GPS. Además dispone de una BBDD MySQL en la que podemos almacenar toda la información que deseemos y comunicarnos si es necesario con la nube.

- Plug and Sense, ver figura 2

Figura 2: Plug and Sense

Estos dispositivos están equipados con 6 conectores a los que podemos acoplar toda una variedad de sensores. Basta con conectar uno nuevo para empezar a

utilizarlo. Si ya tenemos una red con unos dispositivos concretos, podemos conectar uno nuevo y comenzar a utilizarlo de inmediato. Además disponen de placas solares para autoabastecerse y funcionar durante años.

- Waspmote, ver figura 3

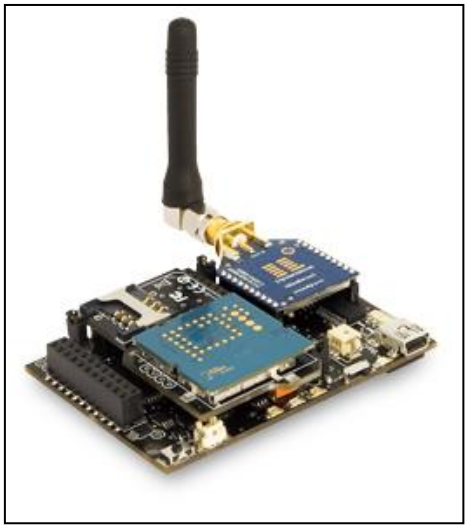


Figura 3: Waspmote

Estas placas se encargarán de recopilar información. Normalmente se equipan con uno o varios sensores y un módulo de comunicaciones. Se suelen desplegar varias placas de este tipo y cada una trabaja de forma autónoma pero con un mismo fin. Funcionalmente son parecidas al Plug and Sense. A pesar de que su puesta en funcionamiento es algo más costosa en tiempo, su tamaño es menor y pueden adaptarse mucho mejor a las necesidades de cada proyecto.

Estas últimas son las que mejor se adaptan a las necesidades del proyecto. Un dispositivo pequeño, de muy bajo consumo y que funcione de manera autónoma.

A este tipo de placas se le pueden acoplar más de 60 sensores distintos, para soluciones como: ciudades inteligentes, control climático, seguridad y emergencias, logística, agricultura y ganadería, domótica, eSalud, etc.

Fase I: comunicando placas

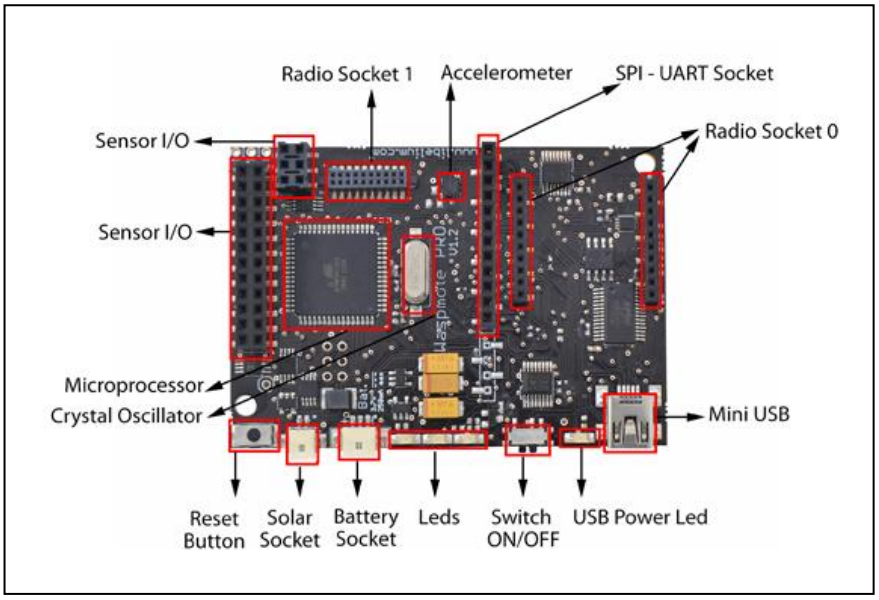


Figura 4: Placa Base Waspmote

Primero debemos familiarizarnos con las placas Waspmote. Como se ve en la figura 4 disponen de varias conexiones para conectar otras placas y antenas; un conector miniUSB para programarlas y monitorizar en fase de pruebas; interruptor ON/OFF; varios LEDs y más elementos que no son de interés para este proyecto.

En Radio Socket 0 ira insertada la antena. Es importante señalar que esta no puede estar conectada cuando queramos cargar un programa en la placa, lo que es un poco engorroso por tener que desenchufar la antena, figura 5, constantemente para reprogramar la placa.

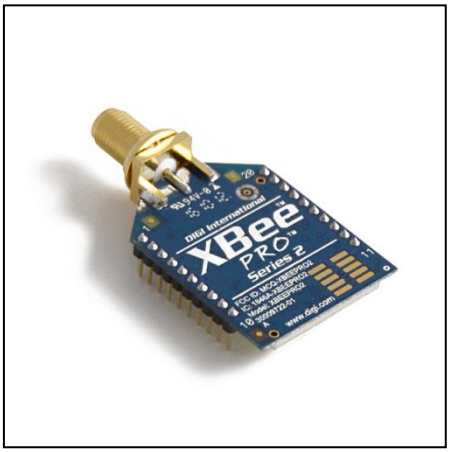


Figura 5: modulo Zigbee

Para comenzar, me dedico a que las placas puedan comunicarse entre sí. Disponemos de antenas con tecnología Zigbee. Este tipo de comunicación inalámbrica viene especificado por el estándar IEEE 802.15.4, que se trata de un estándar de comunicación inalámbrica para redes de área personal. Sus características más reseñables son que se utiliza para transmisiones con una baja tasa de datos y una topología definida de tipo malla preferiblemente. Otra tecnología de características parecidas y con la que suele compararse es Bluetooth. No obstante no son rivales entre sí, pues sus prestaciones varían con mucho. Por ejemplo, Bluetooth sólo permite subredes con 8 nodos mientras que Zigbee admite hasta 255. Y aunque la tasa de transferencia es menor en Zigbee, es suficiente y para este proyecto cubre otras necesidades, como son por ejemplo la de integrar en una misma subred todas las farolas que queramos controlar. Además tiene un consumo muy inferior al de Bluetooth.

Primero se debe decidir el rol de cada una de las antenas. En una topología de red Zigbee debe haber un nodo que actúe como coordinador. Este será el encargado de crear la red y definir sus características: canal, seguridad en las comunicaciones, etc. El resto de nodos pueden ser dispositivos finales o routers. En nuestro caso, no importa demasiado cuál de todas las antenas es la coordinadora.

Para la configuración de las antenas, su fabricante DIGI, nos proporciona el software X-CTU, ver figura 6. Deberemos elegir el tipo de modem/antena, su rol o *function set* y después modificar los parámetros deseados. Para las pruebas podemos dejar los parámetros por defecto, excepto PAN ID. Este es el identificador de la red y debe ser el mismo en todos los dispositivos que queramos que estén bajo la misma red, de lo contrario no conseguiremos que se comuniquen. El tipo de modem viene determinado por la antena que utilicemos, que en nuestro caso será la XB24-ZB. Existen parámetros que pueden hacer que no consigamos establecer comunicación entre antenas, por lo que lo más recomendable es dejar los valores por defecto.

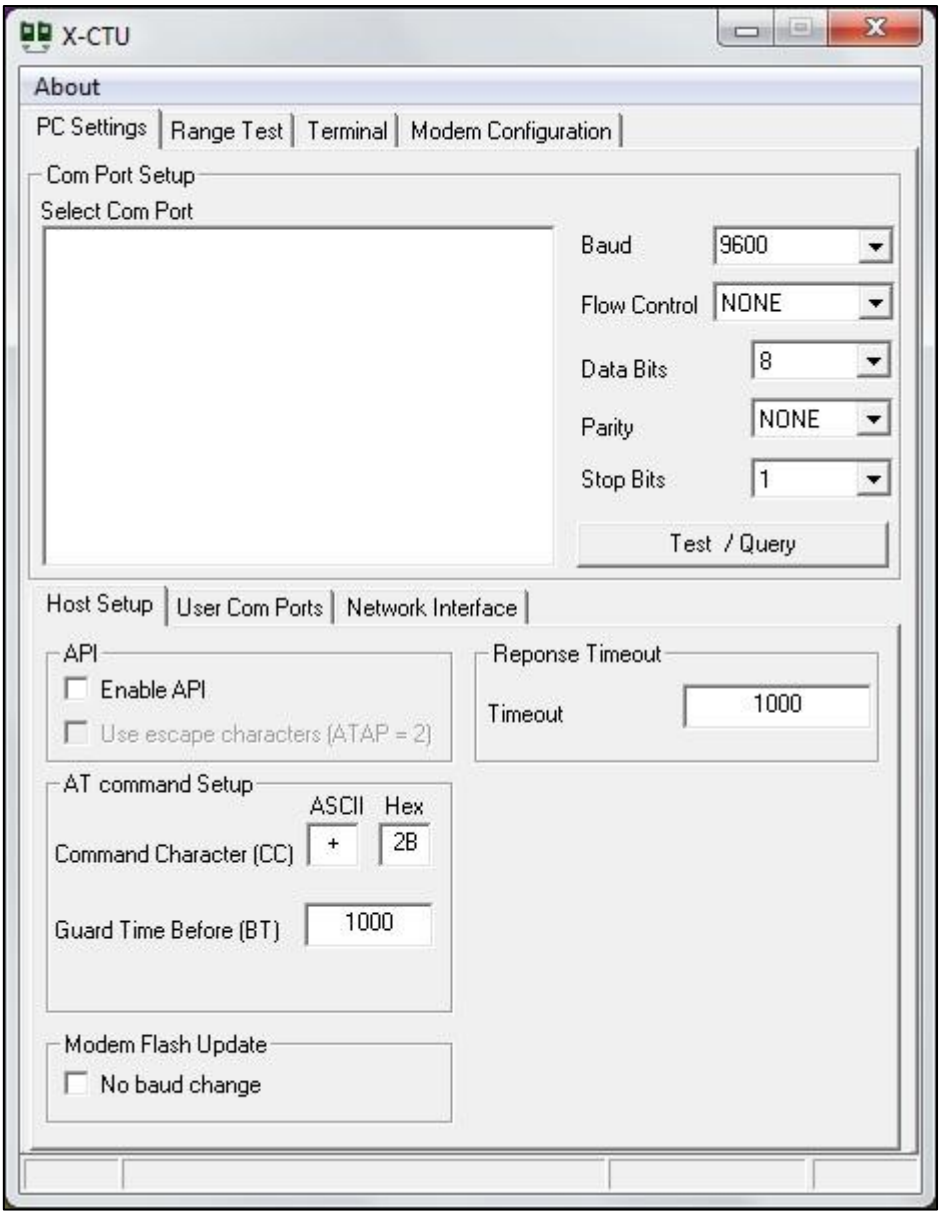


Figura 6: Software X-CTU

Para programar las placas, disponemos de un IDE con todas las funciones necesarias para el desarrollo de sistemas basados en Waspmote. El código se ha de estructurar de la siguiente manera (ver figura 7):

Un primer bloque *setup* donde se inicializarán los componentes del sistema: placas, antena, LEDs con los parámetros que se deseen. Este bloque de código sólo se ejecutará cuando se inicie la placa.

Después viene un segundo bloque *loop* que se ejecutará indefinidamente. La idea es que el sistema repita constantemente las instrucciones programadas.

También se pueden declarar variables y se puede hacer en cualquier parte del programa, aunque por cuestión de orden se deben declarar al comienzo de este, antes del bloque *setup*.

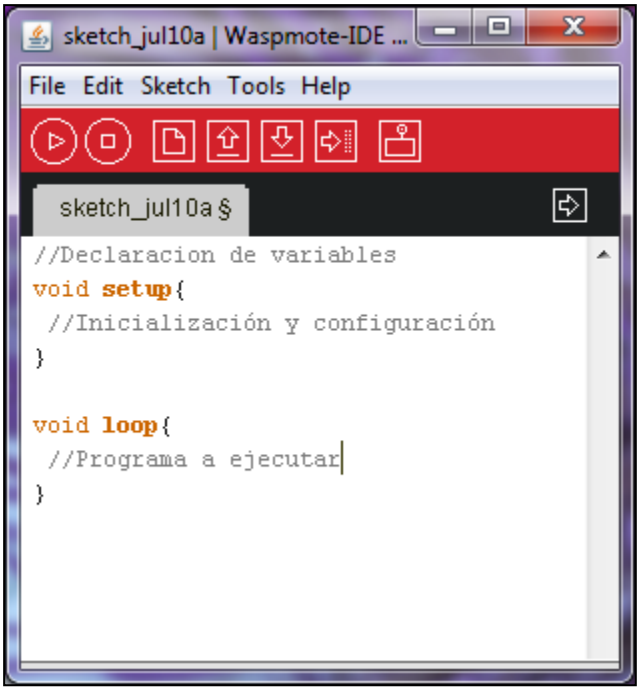


Figura 7: estructura código fuente Waspmote

Veamos en el código usado qué instrucciones se deben utilizar y su función.

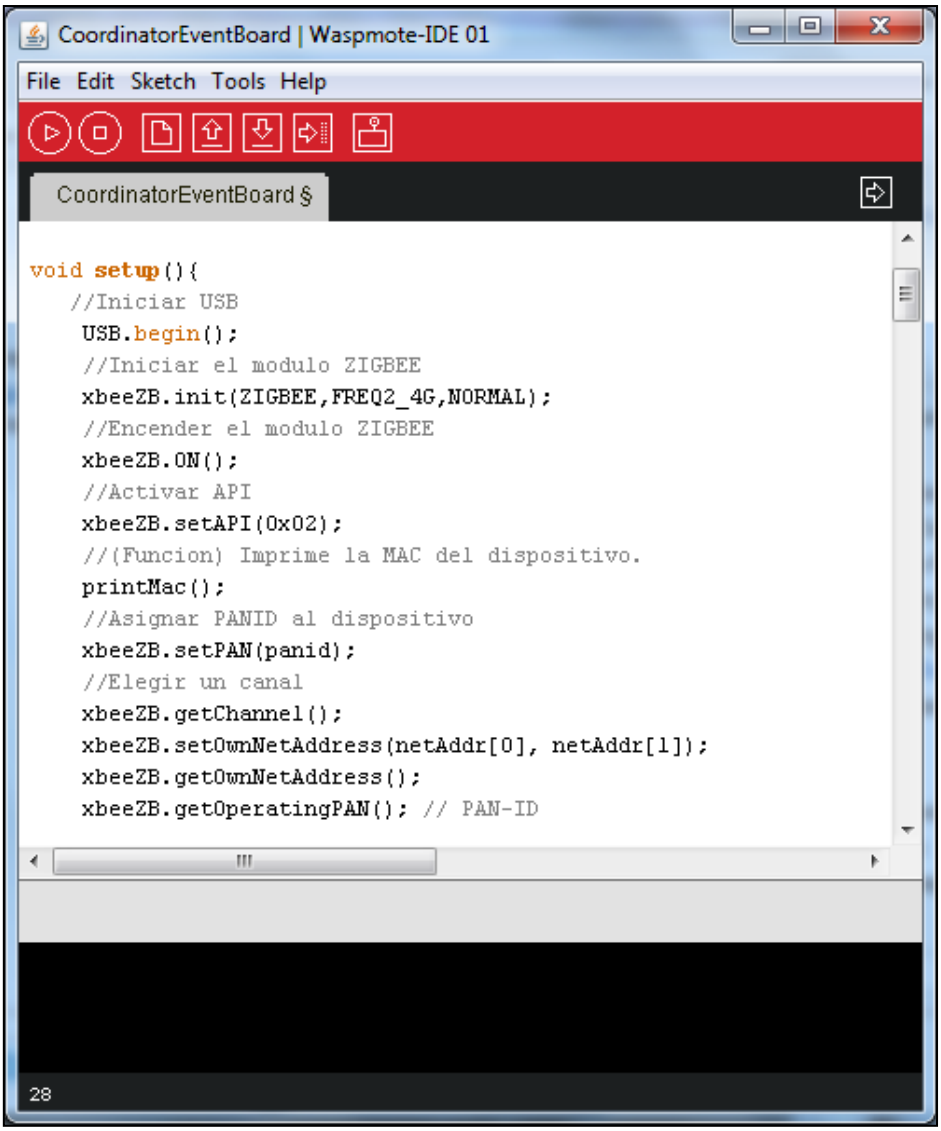


Figura 8: bloque setup

En la figura 8, vemos el bloque setup, donde inicializamos diferentes funcionalidades de la propia placa. Veamos las más importantes:

USB.begin(): así iniciamos el puerto USB, para poder imprimir mensajes durante las pruebas y depuración del sistema. Cuando se ponga a punto, podemos eliminar esta y todas las instrucciones que utilicen esta funcionalidad, como lo son USB.println().

xbeeZB.init(): inicializa el módulo de comunicaciones de acuerdo a los parámetros que le indiquemos. Nuestro dispositivo trabaja con tecnología Zigbee, a 2.4Ghz y en su versión Normal. Aunque, las antenas que usamos son de la versión PRO, una versión posterior mejorada y con mayor alcance, la configuración de ambos módulos es idéntica y las funciones para manejarlos las mismas, por lo que podemos usar el parámetro NORMAL o PRO indistintamente.

xbeeZB.ON(): tras inicializar el módulo, es necesario indicarle que se encienda. Bastará con esta simple instrucción.

xbeeZB.SetPANID(uint8_t): aunque este parámetro se puede configurar con el software X-CTU, es más flexible hacerlo en el código, porque usando la funcionalidad de programación sobre el aire (Over the Air Programming) y cambiando el código de las placas, no hace falta tener acceso físico al dispositivo para cambiar este y otros parámetros. Recordar que PANID debe ser el mismo para todos los dispositivos que queramos intercomunicar en una misma red.

Una vez configurados estos parámetros básicos, si no ha habido ningún error en el proceso, el sistema pasará a ejecutar el bloque *loop*.

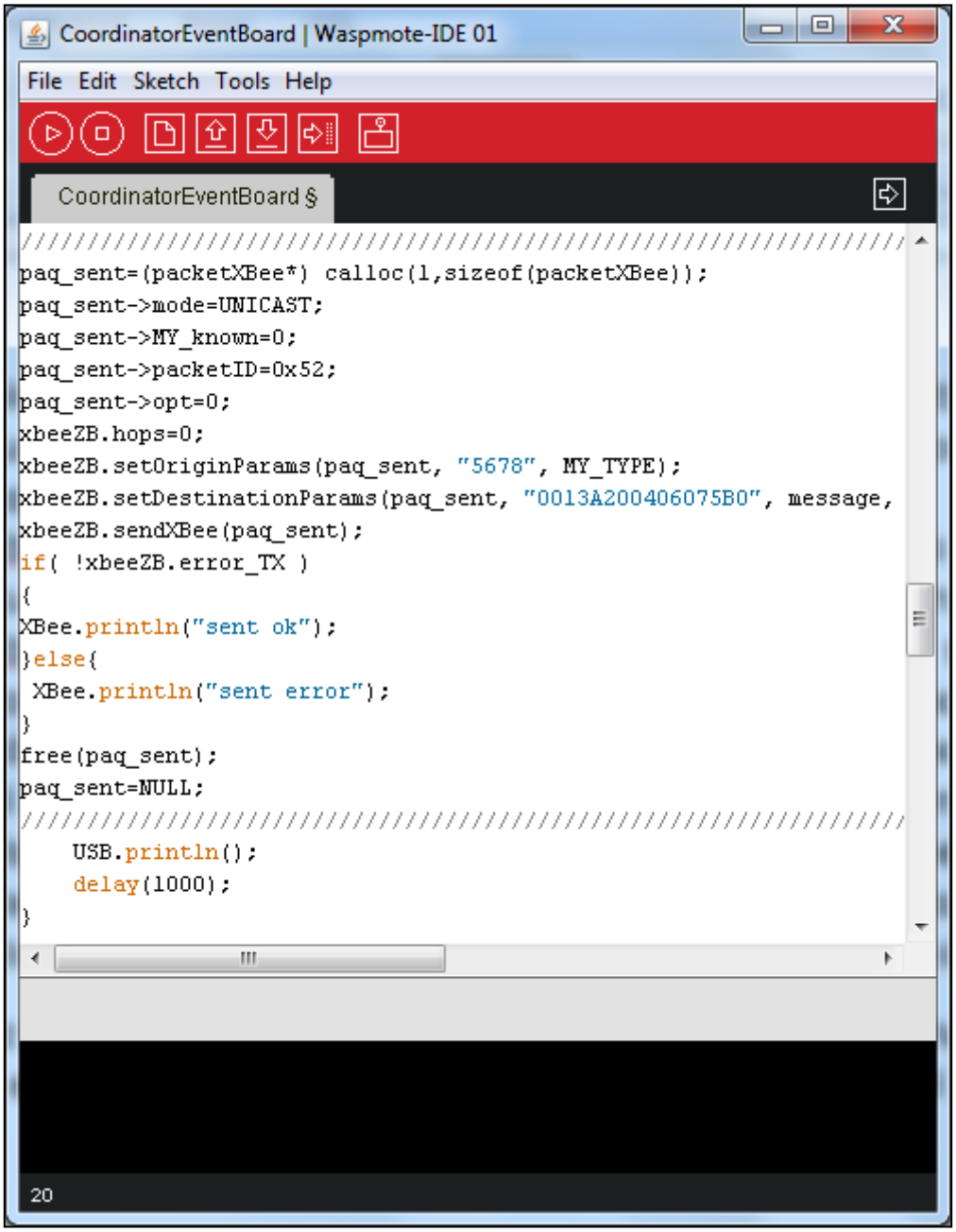


Figura 9: preparación paquete Zigbee

Para el envío de mensajes, existen funciones definidas que nos facilitarán mucho nuestro trabajo. Básicamente debemos construir un objeto de tipo *XBeePacket*. A este objeto le diremos si es para un único destino, UNICAST o para todos BROADCAST, si el origen es conocido y su identificador entre otros, tal y como vemos en las primeras 5 instrucciones en la figura 9. Sin embargo, los datos que queremos enviar o el destinatario no se le indican de forma directa al paquete, sino que lo hará el módulo de comunicación.

xbeeZB.setDestinationParams(packetXBee, destino, datos, id_origen, tipo_datos): esta es la instrucción clave de la comunicación. Al objeto *packetXBee* que hayamos preparado, se le indicará:

- Destino: puede ser la dirección MAC o la dirección de red, si se la hemos asignado. En este caso usamos las direcciones MAC y así nos evitamos tener que configurar otro parámetro.
- Datos: los datos se añaden al final del paquete. Deben ser del tipo char* o uint8_t*. El tipo uint8_t puede compararse con el int o integer de otros lenguajes de programación.
- Id_origen: se puede indicar que el nodo de origen sea identificado por su MAC o por su dirección de red. En este proyecto no utilizamos direcciones de red, por lo que será su dirección MAC.
- DATA_ABSOLUTE: es un parámetro que indica que los parámetros indicados se han de incluir en el paquete.

Tras preparar el paquete, sólo queda enviarlo mediante la siguiente función:

xbeeZB.sendXBee(paq_sent): indicando como único parámetro el paquete que hayamos preparado previamente.

Tras esto, *loop* volverá a ejecutar estas instrucciones de inmediato, por lo que se suele utilizar una instrucción para pausar un momento la ejecución:

delay(1000): esta instrucción detendrá la ejecución por tantos milisegundos como indiquemos.

Hasta aquí la parte de código en la que un nodo inicia la comunicación. Necesitamos que otro nodo, el destinatario del paquete que enviamos este funcionando y a la escucha de los mensajes que pueda recibir.

El código varía poco: el bloque *setup* es exactamente igual, puesto que ambas antenas deben configurarse de idéntica manera. Sólo variarán en algunos parámetros, como la

dirección de red, si procede. Veamos pues cómo el bloque *loop* espera la recepción de mensajes.

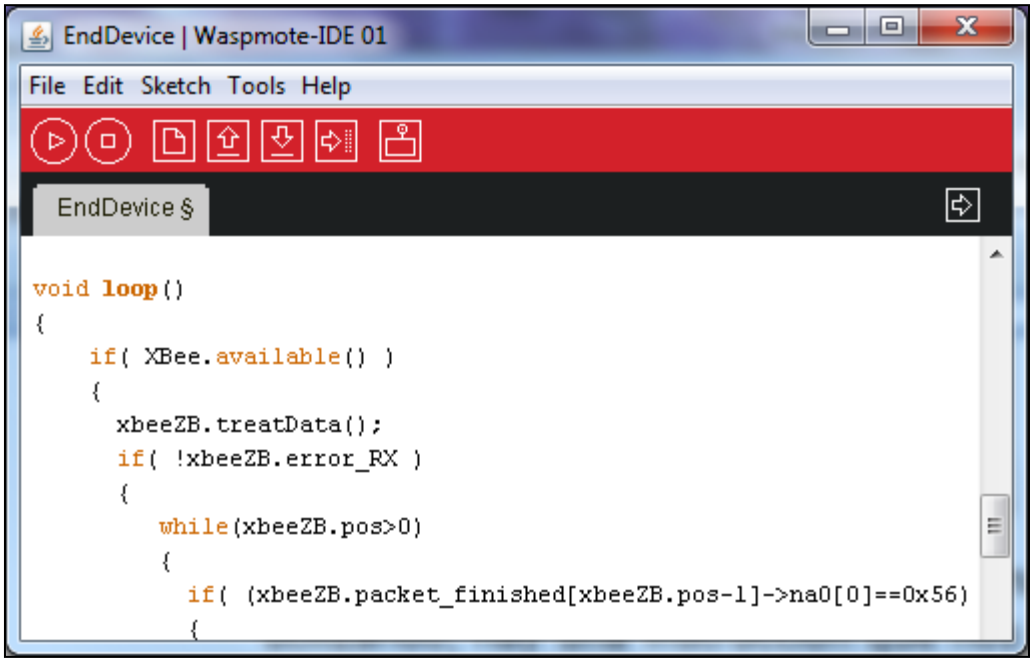


Figura 10: comprobando si hay datos disponibles en la antena

Independientemente de la tecnología inalámbrica que estemos utilizando, hay una instrucción que nos permitirá saber si hay algún dato disponible: XBee.available(). De haberlo, el tratamiento de los datos sí que habrá que hacerlo con funciones propias de cada tecnología. Digamos que la función XBee.available() avisa de que hay datos, pero los datos no se tratan igual si se han enviado mediante Zigbee, Bluetooth, WIFI, etc. Así pues, en Zigbee usaremos xbeeZB.treatData() y los datos contenidos en el paquete recibido se almacenaran en una variable llamada packet_finished habilitada para poder acceder a ellos de forma fácil.

Y a partir de aquí, quedaría el tratamiento de los datos, y envío de mensajes si procede.

Como se puede comprobar la comunicación entre dispositivos está bastante depurada y bastan unas pocas instrucciones para ello. Por supuesto hay que tener cuidado a la hora de configurar todos los dispositivos y conocer su dirección para hacerles llegar los datos.

Fase II: recopilando datos de los sensores

Para acoplar los sensores de presencia y luminosidad necesitamos una placa de eventos como la de la figura 11, que ira acoplada a la placa base.

Figura 11: placa de eventos

En ella apreciamos el sensor de presencia, la semiesfera blanca y el de luminosidad en primer plano entre un bloque dorado y otro negro, entre otros. Este último es el sensor de temperatura, que en lugar de colocarlo en esta placa, irá en la placa de gases.

Para poder usar estos dispositivos también tenemos que inicializar algunos parámetros en el bloque *setup*, ver figura 12.

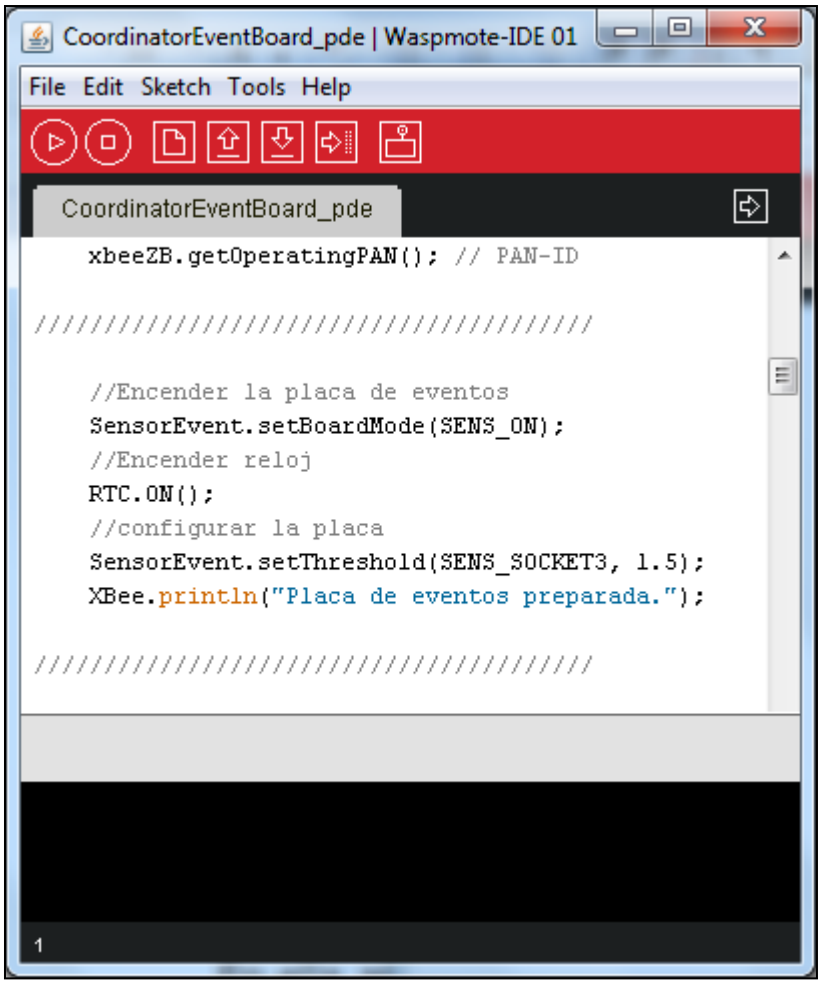


Figura 12: configurando sensores

SensorEvent.setBoardMode(SENS_ON) activa la placa y la prepara para que pueda mandar datos de los sensores que tiene conectada.

SensorEvent.setThreshold(SENS_SOCKETX,volt) activa el socket indicado y configura el valor de voltaje para el cual, sobrepasándolo causará una interrupción. Independientemente del tipo de sensor que sea, el valor que se puede recoger de estos es un voltaje que va desde 0V hasta 3.3V.

Una vez configurada la placa, solo queda leer los valores. A pesar de que en el bloque *setup* parece que sólo vamos a utilizar el socket 3, en realidad hay otro socket que también va a ser utilizado y que no necesita ser configurado: el nº 7, al que conectaremos el sensor de presencia.

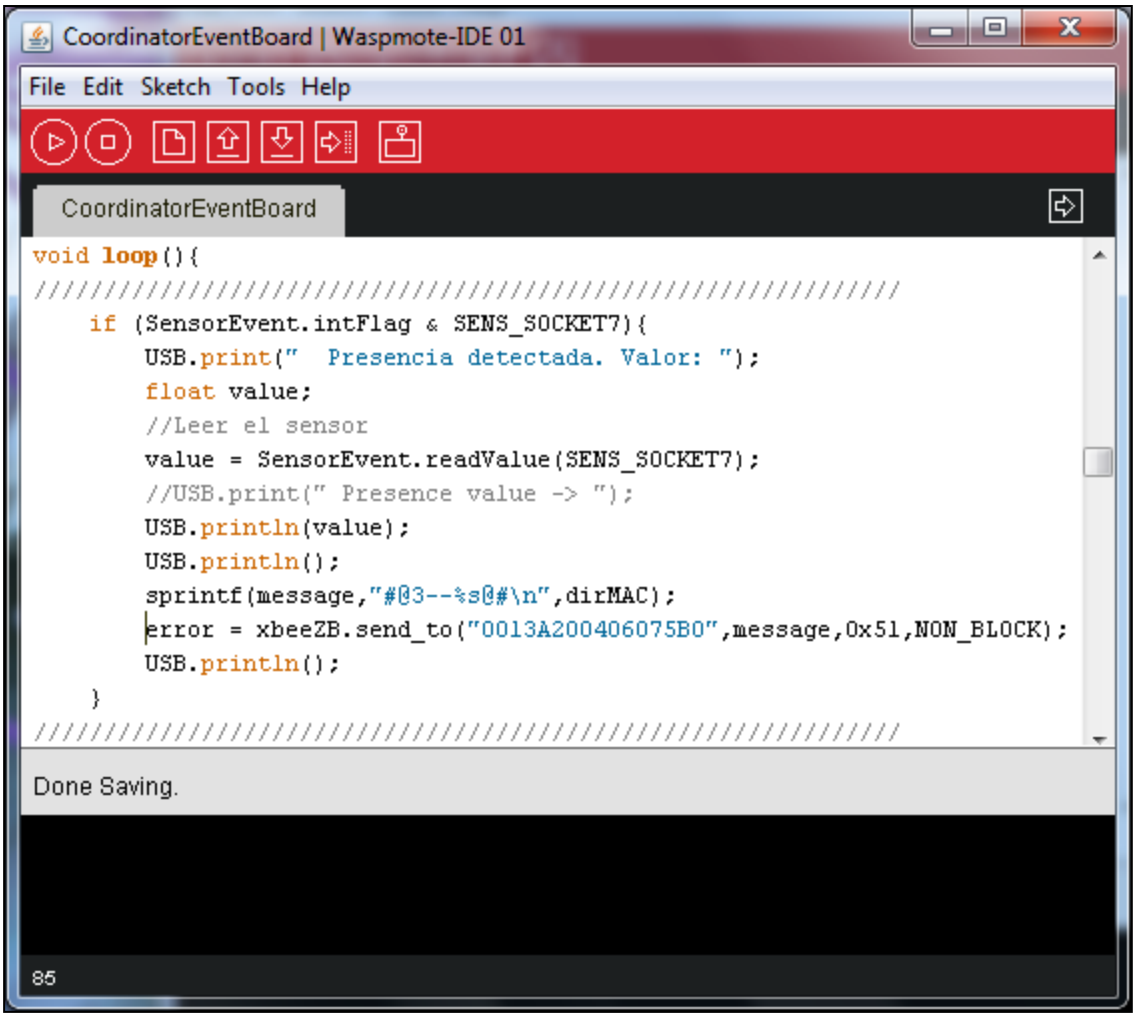


Figura 13: comprobando presencia

El sensor de presencia, no necesita de configuración alguna para poder ser utilizado. Una vez conectado, provocará una interrupción, marcando su *flag* correspondiente y esta debe ser detectada en el código. Así, con una simple instrucción condicional, preguntaremos si el flag activado corresponde al socket 7. Podemos leer el valor que retorna el sensor mediante una simple función:

SensorEvent.readValue(SENS_SOCKET7)

Aunque no es necesario. Esta lectura devolverá 0 si no se detecta presencia y 1 si ha sido detectada. Como digo, al detectarse provocará una interrupción, por lo que el valor leído no nos aporta información adicional.

La lectura del valor de luminosidad tiene algunas peculiaridades. Como hemos comentado, los sensores no devuelven un valor que se corresponda con la magnitud asociada, sino un valor de voltaje entre 0 y 3.3V. Este ha de ser convertido para obtener un valor en lux con las formulas que se ven en la figura 14

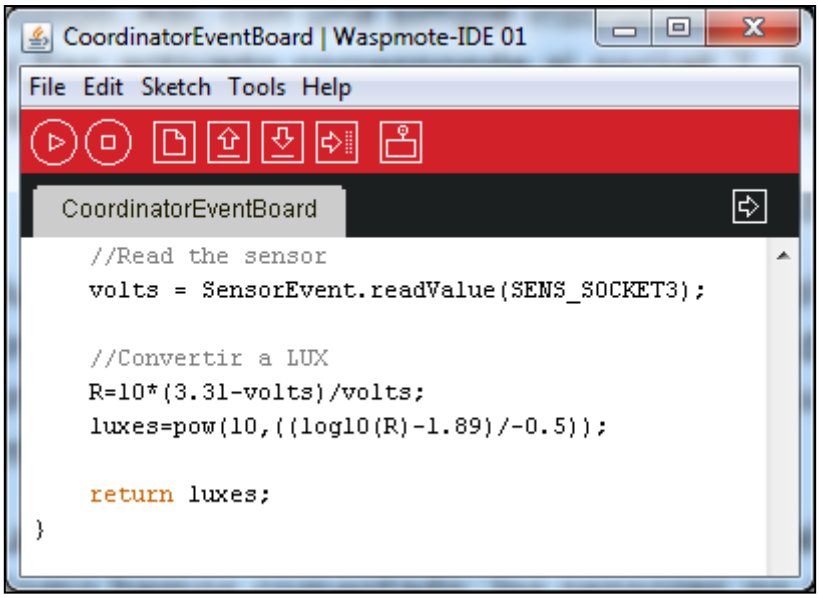


Figura 14: leyendo sensor de luminosidad

Como puede apreciarse, el proceso de recogida del valor es exactamente el mismo que en el caso del sensor de presencia. El valor obtenido en este caso son volts y con las formulas que proporciona el fabricante

$$\mathrm{R} = 10\mathrm{x}(3.31 - \mathrm{volts})/\mathrm{volts}$$

$$\mathrm{lux} = 10^{((\log_{10}\mathrm{R})-1.89)/-0.5}$$

obtenemos los luxes que hay en el ambiente en ese momento.

En el momento de configurar este socket, podemos determinar el umbral para el que queremos que provoque una interrupción, pero he considerado más oportuno recopilar información de la luminancia y trabajar directamente con ese valor. En el caso de que queramos modificar el umbral mínimo para el cual queremos activar el sistema, basta con modificar el valor en lux y no tener que calcular de nuevo qué valor en volts se corresponde con el de lux deseado.

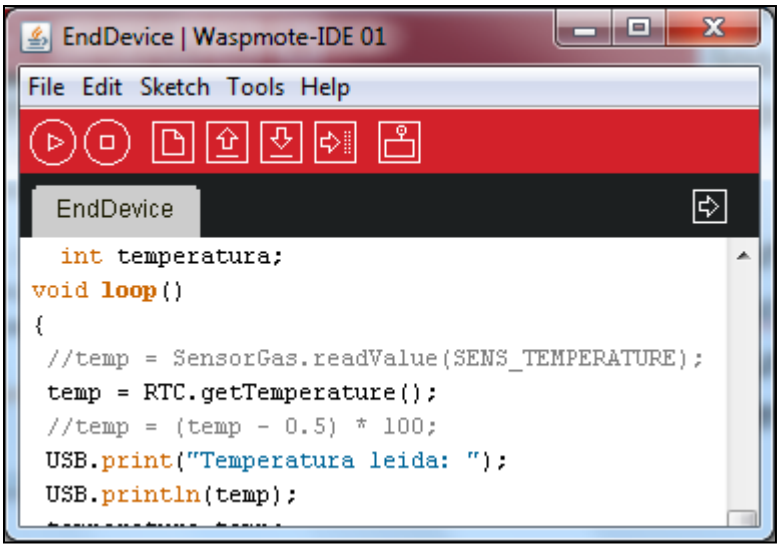


Figura 15: leyendo la temperatura

La temperatura es otro valor que hemos decidido tomar. Aunque no es tan decisivo a la hora de decidir cuánta potencia se debe transmitir a las farolas, puede que más adelante nos sea útil. El sensor utilizado (ver figura 8) al igual que el sensor de luminosidad devuelve un valor en volts, el cual debemos convertir a la magnitud de temperatura que deseemos. Cabe mencionar que los valores devueltos no eran correctos usando la placa de eventos, por lo que también probé a recoger valores en la placa de gases. Como en ninguna de las dos placas parecía estar leyendo correctamente la temperatura, me decanté por utilizar el sensor de temperatura que tiene integrada la placa base. Según información del foro del fabricante, este es menos preciso pero en las pruebas arrojaba valores más cercanos a la realidad.

Al ser un sensor integrado, existe una función que devuelve la temperatura directamente, sin tener que convertir ningún otro valor.

RTC.getTemperature();

Fase III: decidiendo cuándo y cuánto encender las farolas

Ya sabemos cómo comunicar los dispositivos y recoger valores de los sensores de que disponemos. En un principio el sistema se iba a basar sólo en los valores de presencia y luminosidad. De esta manera sólo habría que distribuir el valor de luminosidad por todas las placas y estas, en función de dicho valor y de su sensor de presencia asociado actuarían directamente sobre su luminaria. La comunicación se haría entre placas únicamente.

Se vio la posibilidad de mejorar el sistema, montando un sistema experto que lo dotara de cierta inteligencia y así mejorar las prestaciones del mismo. No obstante, la capacidad de procesamiento de las placas es limitada y se optó por montar el sistema en un PC de sobremesa. Esta decisión, obligaría a cambiar el modo en que se comunicaban las placas. A partir de ahora, la información de luminancia, presencia y temperatura se recogerá en un punto central, el PC, se almacenará en una base de datos y tras el tratamiento de la información, se mandará la orden oportuna de encendido de forma individual. Esto supone trabajar en dos frentes:

1. Reordenar el código de las placas para que atiendan a la nueva forma de trabajo
2. Montar el sistema en el PC.

Este segundo paso supone introducir sistemas que hasta ahora no se habían visto. Explicaré los pasos que seguí en su implementación:

COMUNICACIÓN: la tecnología sigue siendo la misma, Zigbee, pero para que un PC pueda manejar estas antenas, es necesario un adaptador como el de la figura 16

Figura 16: Gateway

La antena se encaja en el adaptador, Gateway y este a cualquier puerto USB del ordenador.

BBDD: se ha instalado el gestor de bases de datos MySQL, por ser gratuito y fácil de manejar. En el disponemos de dos tablas: un log donde se registra cada evento y otro donde tenemos identificadas las placas que funcionan en el sistema. Para cada evento guardamos los siguientes datos:

- Tipo de evento: 1=luminancia; 2=temperatura; 3=presencia
- Valor: para la luminancia y temperatura sus valores y en el caso de presencia el identificador de la placa.

Datos extra: dispondremos de una tabla con la hora del orto y ocaso de todos y cada uno de los días del año. De esta manera, podremos programar mejor las horas en las que el sistema estará encendido y ahorrar al máximo en las baterías que lo alimentan. Sólo el hecho de monitorizar la luminancia supone un gasto de energía y si sabemos entre qué horas el sol va a estar fuera, podemos definir un periodo de tiempo en el que todas las placas entren en un estado de muy bajo consumo, casi nulo.

Todo esto será gestionado por un programa en JAVA y un API para manejar la antena que nos proporciona DIGI.

Problemas surgidos

Esta última fase no se ha podido desarrollar completamente. El tratamiento de los datos no ha sido cuestión difícil, al fin y al cabo es tratamiento de cadenas y acceso a un archivo Excel. Hubo cierto problema con el manejo del GBD. Al intentar hacer uso de este, saltaba un error de log, al que no le di mayor importancia hasta que descubrí que no era capaz ni de acceder a las tablas. Tras una larga búsqueda en internet descubrí que el GBD MYSQL utiliza un servidor web a modo de log, donde va guardando registro de toda la actividad. Aunque seguramente se podría configurar MYSQL para que no registrara dicha actividad en un servidor web, al buscar el problema, me topé con la solución y opté por instalar un servidor web para solventar el problema y no retrasar más el trabajo. No obstante este log no estaba ni planteado en el diseño del sistema y se puede prescindir de él.

Sin duda, el mayor problema y que se ha llevado más de ¾ partes del tiempo dedicado a todo el proyecto es el tema de las comunicaciones. En la FASE I invertí algo más de un mes de trabajo sin conseguir avanzar nada. Los códigos de ejemplo que facilita el fabricante no funcionaban; instrucciones simples de configuración de la antena tampoco; se probaron los códigos de ejemplo de todas las tecnologías inalámbricas que prevé el fabricante y tampoco se obtuvo ningún resultado positivo. Llegamos a plantearnos si las antenas en realidad no funcionaban con tecnología Zigbee, que tras consultar con el servicio técnico nos confirmaron que sí.

Algunos códigos de ejemplo no estaban bien construidos, tenían fallos etc. Y me centré única y exclusivamente en encontrar algún fallo en el programa, lo que me llevó a actualizar las librerías del programa, a utilizar una versión posterior del programa incluso a probar códigos que gente con problemas había colgado en el foro del fabricante y que tras solucionar el problema colgaba su código corregido y en ningún caso avancé en mandar ni recibir ni un solo dato.

Descartado el problema del código, comencé a indagar en la configuración de las antenas. Como ya hemos visto, el software X-CTU nos permite configurar los valores de los módulos de las antenas. Conectamos el módulo al Gateway y este a un puerto serie, que seleccionaremos en el programa para que sepa en cuál debe trabajar.

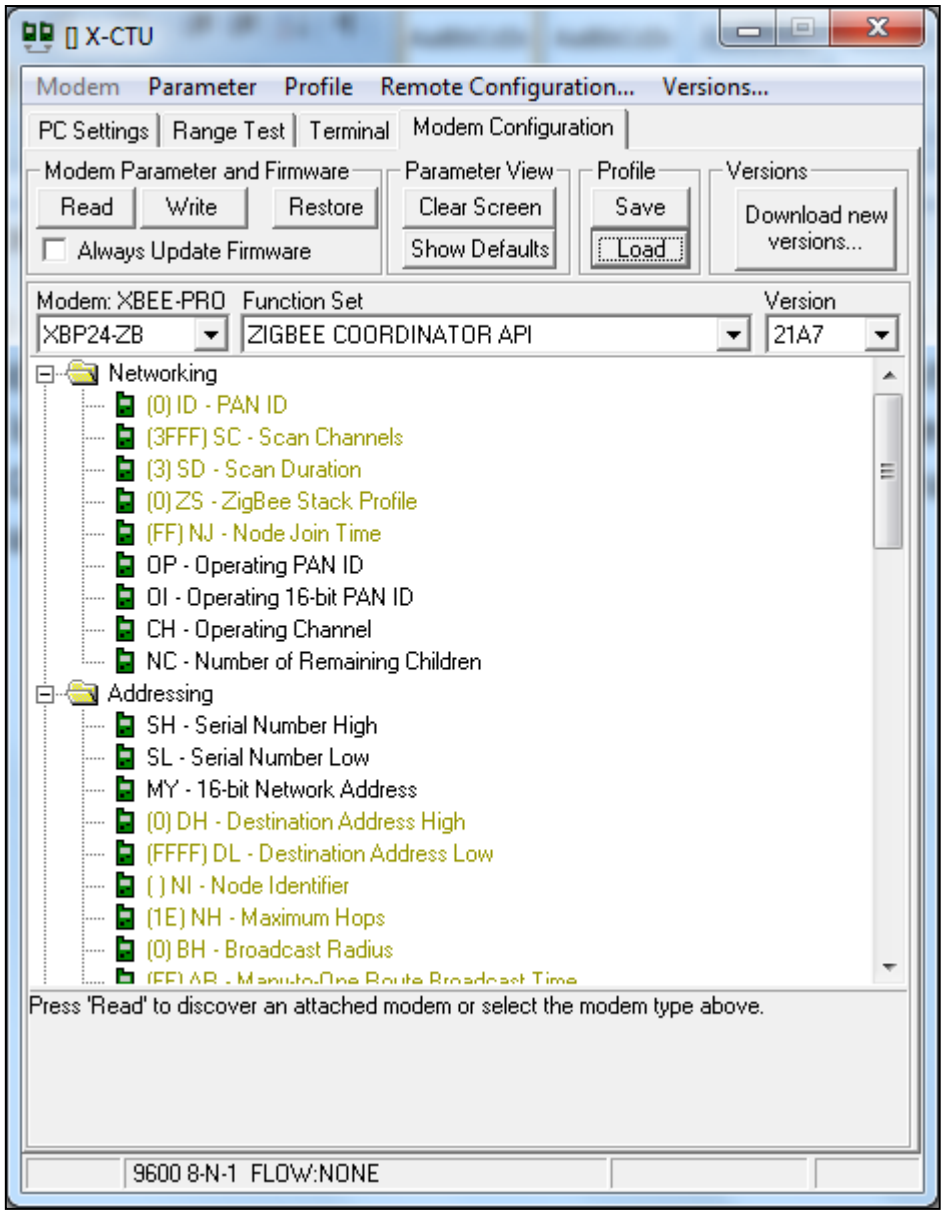


Figura 17: parámetros configurables X-CTU

En la figura 17 podemos ver la configuración elegida para la antena que va a actuar como coordinadora.

En un principio todas las antenas estaban configuradas como router. Aunque esto no debería suponer mayor problema para establecer comunicación, según las topologías que admite Zigbee, debe existir uno y solo un coordinador, que será quien configure la red y todos los parámetros asociados. Así pues, seleccioné el *Function Set* del coordinador y así se cargaron los valores por defecto (ver figura 17). Como puede apreciarse, el PANID está a 0. Esto indica que la antena seleccionará uno al azar. Podemos indicarle aquí un número arbitrario, pero para no complicar más las cosas, decidí dejarlo tal cual. Cuando otra antena se active, buscará redes que estén a su alcance y se unirá a esta.

Otro parámetro al que hay que prestarle atención es al *BaudRate* (velocidad de transmisión, tasa de bits). Todas las antenas han de funcionar con la misma tasa de bits o de lo contrario tampoco funcionarán correctamente. A este respecto, la tasa de bits por defecto es 9600, pero en diferentes foros se recomienda 38400. Para evitar problemas, también hay que cambiar la velocidad en la barra de estado del programa X-CTU, que por defecto estará en 9600 también (ver figura 17). Aunque a priori no causa problemas a la hora de cargar la configuración, es mejor que ambas velocidades coincidan.

En la topología que describe el fabricante hay otros dos tipos de nodos: routers y dispositivos finales o *end device*. En mi caso, he configurado dos como *end device*. Basta con seleccionar en *Function Set* esta opción y fijarse que tanto en PANID como en *Baudrate*,. Coinciden con el del resto de antenas configuradas.

Tras reconfigurar las antenas, los códigos de ejemplo del fabricante seguían sin funcionar. Opté por usar una versión del programa con el núcleo modificado, que ya hubieran utilizado otros compañeros de la carrera para sus proyectos, y comprobé que en este caso la comunicación se establecía y se podían mandar mensajes de una antena a otra. Por alguna razón las librerías facilitadas por el fabricante no funcionan correctamente con las antenas. En el foro no encontré ningún hilo en el que tuvieran problemas de este tipo.

Las placas y las antenas son de fabricantes distintos. Cada uno se configura con un software distinto y aunque Waspmote lo prevé y dispone de funciones para modificar los parámetros de las antenas, estas funciones no modificaban los valores correspondientes en tiempo de ejecución. Hay un manual para el manejo de las placas y otro para la configuración de las antenas pero al menos yo, echo en falta uno en el que indique que para cierta configuración de la antena, se debe proceder de una u otra forma a la hora de intercomunicarlas. Como prueba, utilizando la configuración por defecto en las antenas y los códigos de ejemplo no se establece comunicación entre ellas.

Y como problema final, otro de comunicación. Descubrí que a pesar de que las antenas eran capaces de enviar y recibir mensajes, la función que lo hacía devolvía un código de error. Después de un mes sin conseguir que se comunicaran lo obvié, puesto que al menos podía mandar los datos de una placa a otra. Pero esto me daría problemas cuando quise tratar estos mensajes en el ordenador. DIGI dispone de unas librerías

para JAVA que nos permiten crear paquetes, enviar y recibir datos por el puerto serie. Leyendo lo que llegaba del puerto serie era capaz de extraer los datos de luminancia, temperatura y saber en qué placa se había detectado presencia. Cuando quise utilizar las funciones de DIGI para manejar los paquetes, parecía no recibir ninguno. El error que se mostraba al enviar el paquete, debe estar relacionado con el *checksum* del paquete y al no ser correcto se desechaba directamente al llegar al PC.

Hice varias pruebas de envío de paquetes desde JAVA a alguna de las placas pero no conseguí ningún resultado. Al crear el paquete, parecía ir todo correcto. Pero al ejecutar la instrucción de envío, esta no hacía nada.

Leer, había conseguido leer, aunque fuera byte a byte desde el puerto serie, por lo que otra solución para conseguir enviar un paquete podría pasar por preparar el paquete y escribirlo directamente en el puerto serie, a la inversa de como recibía los paquetes procedentes de las placas. De nuevo, no conseguí ningún avance en este sentido.

Finalmente y para que hubiera un prototipo funcional, he decidido dejar de lado, al menos de momento y preparar un sistema con sólo las placas de Waspmote. Se distribuirá el valor de luminosidad entre todas las placas y estas decidirán de forma individual si encender más o menos las farolas. A falta de luminarias para la simulación, se usarán los led incorporados en las placas. La topología utilizada para la comunicación entre placas es de tipo Mesh.

PRUEBAS, VALIDACIÓN Y PUESTA EN MARCHA

PRUEBAS

Las pruebas del sistema se han realizado en un laboratorio de la Universidad Pública de Navarra, con los equipos allí disponibles.

Al no poder integrar del todo la red de sensores y el sistema de tratamiento de datos en un PC, las pruebas se han dividido en dos fases:

- Probar la monitorización y comunicación entre los sensores
- Probar el funcionamiento del sistema sobre el PC de almacenamiento de datos.

Para la primera, se prepararon dos placas, que serían las encargadas de recopilar información de los sensores y se conectaron al ordenador para verificar que las placas recogían la información y eran capaces de transmitirla. A través de un monitor del puerto serie, se podía comprobar cómo los paquetes llegaban.

A la vez, se probaron los sensores para ver que funcionaban correctamente y según los datos variaban, también los datos que transmitían las placas.

- Sensor de luminancia: con el sensor destapado, arrojaba datos alrededor de los 350lux, variando levemente. Si se proyectaba una sombra sobre este el valor bajaba hasta los 100lux y cubriendo totalmente el sensor el valor era de 0. Usando un LED de un móvil y apuntando su luz al sensor, se alcanzaban valores de 3000lux. Todo cambio se recibía correctamente en la antena que se preparó a tal efecto.
- Sensor de temperatura: se probaron tanto el sensor integrado como el que se podía acoplar a la placa de gases. Este último devolvía valores que oscilaban entre -45ºC y -30ºC, sin provocar ningún cambio de temperatura sobre él. El valor devuelto no era para nada real y si no fuera por la gran variación entre los valores podría pensarse que es un error en la conversión del valor que devuelve el sensor; pero no es el caso. El sensor integrado, devolvía un valor de 17ºC, que aunque menor que la temperatura real, se mantenía estable y respondía a los cambios de temperatura inducidos. Igualmente estos cambios se podían observar en la antena receptora.

- Sensor de presencia: este sensor, a diferencia de los dos anteriores, no manda periódicamente mensajes sino que lo hace cuando se detecta una presencia. Para comprobar su correcto funcionamiento, bastaba con dejar el sensor en reposo, pasar la mano y comprobar que este enviaba el mensaje a la antena receptora.

Hasta aquí, el proceso de monitorización y comunicaciones parece correcto.

En la segunda fase se probó cómo la información era recopilada en el PC y almacenada en una BBDD local. Se aprovecharon las mismas pruebas que en la primera fase para ver como cada registro cambiaba el valor de luminosidad, temperatura o placa que había detectado la presencia.

PUESTA EN MARCHA

Los pasos a seguir para encender el sistema son sencillos y los explico de forma clara y concisa a continuación:

Primero se han de disponer de todos los elementos para construir cada nodo: placa, batería, pila, modulo de comunicación y antena y placa de eventos con sus respectivos sensores (presencia, temperatura y/o luminancia).

Antes de conectar todos los dispositivos se han de programar tanto las antenas como las placas. En una de las antenas se deberá cargar el perfil de *coordinator* y en el resto la de *enddevice (adjuntos)*, todo con el programa X-CTU y con la ayuda del adaptador Gateway.

Para programar las placas, es necesario conectar las placas al ordenador a través de un cable USB y sin tener las antenas conectadas. Con la ayuda del IDE que nos proporciona Libelium, cargaremos el programa correspondiente. En la placa que tenga el sensor de luminancia el programa llamado *coordinator* y en el resto el programa *enddevice*. Para configurar las placas, estas deben estar encendidas, con el interruptor lo más alejado del conector USB. Una vez cargado el programa, apagaremos la placa, conectaremos la antena y placa y podremos conectarla.

Quizás sea necesario cambiar algún parámetro en los programas, como es la dirección de la antena a la que se desea enviar la información. Habrá que decidir cuál de ellas será la que reciba datos de todas las demás e indicar su dirección MAC en los programas de las placas antes de cargarlos.

Hasta aquí todo para que se pueda comprobar el funcionamiento básico del sistema. Se puede actuar sobre los sensores y los LED programables de cada placa deberían encenderse más, en función de la "necesidad" de iluminación que se requiera.

CONCLUSIONES Y LINEAS FUTURAS

Con este proyecto he descubierto las posibilidades que tienen desarrolladores a la hora de montar sistemas de monitorización de forma fácil y barata. Aunque en mi caso parecía tener todo en mi contra y me ha resultado extremadamente difícil manejar estas placas, la cantidad de proyectos que se han desarrollado con esta tecnología avalan su flexibilidad a la hora de diseñarlos y su bajo coste y accesibilidad.

Estas soluciones modulares permiten desarrollar sistemas de monitorización integrales a partir de elementos prefabricados y según el programa que se cargue poder monitorizar las calles, parkings, bosques o casi cualquier otra cosa, gracias sobre todo a la miniaturización de estos sistemas, a las comunicaciones inalámbricas y al bajo consumo.

En la tarea de monitorizar los espacios públicos para adecuar la iluminación de las calles, a falta de pruebas reales, parecen ser una gran ayuda con el objetivo de reducir los costes, manteniendo o incluso aumentando la seguridad y ofreciendo un nivel de confort más que aceptable.

Personalmente he encontrado muy gratificante trabajar en un proyecto que nada tiene que ver con ninguna de las asignaturas que hemos visto en la titulación. Plantear una problemática e intentar ofrecer la mejor solución posible y que además esta tenga una aplicación real y actual motiva aún más a la hora de realizar el proyecto. Creo que las soluciones basadas en esta tecnología y placas, van a tener un auge en muchos ámbitos de nuestra vida, sobre todo en monitorización y control del entorno.

Como líneas de trabajo sobre las que seguir desarrollando este proyecto podrían seguir tres caminos distintos:

1. Seguir con la integración del sistema en un sistema experto en un PC y controlado de forma centralizada.
2. Desarrollar un sistema distribuído que se instale en las placas.
3. Probar a utilizar otro dispositivo de la familia Libelium, como es Meshlium y utilizarlo como dispositivo centralizador.

En cualquiera de los tres desarrollos existen algunos puntos que sería interesante pensar:

- Detección de fallos en el sistema y alarmas: fallo de una placa, luminaria fundida, funcionamiento incorrecto, etc.
- Iluminación inteligente y predictiva: detectar los objetos en movimiento y sentido de movimiento y aumentar la iluminación posterior en función de su velocidad.
- Iluminación viaria más concreta: pasos de cebra y cruces, etc.

BIBLIOGRAFÍA

[1] Libelium <http://www.libelium.com>

[2] CookingHacks <http://www.cooking-hacks.com>

[3] DIGI <http://www.digi.com/>

[4] API DIGI <https://code.google.com/p/xbee-api>

[5] Arduteka <http://www.arduteka.com/2013/01/waspmote-guia-de-iniciacion-empezando-desde-cero>

[6] Zigbee Alliance <http://www.zigbee.org>

[7] Real Decreto 1890/2008 sobre el Reglamento de eficiencia energética en instalaciones de alumbrado exterior y sus Instrucciones técnicas complementarias

[8] Guía Técnica de Eficiencia Energética en Iluminación. Alumbrado Público. IDEA. Disponible en <http://www.idae.es/index.php/mod.documentos/mem.descarga?file=/documentos_GT_EE_iluminacion_Alumbrado_Publico_9a40dc27.pdf>

[9] Requerimientos Técnicos exigibles para luminarias con tecnología LED de alumbrado exterior. Disponible en <http://www.idae.es/index.php/mod.documentos/mem.descarga?file=/documentos_Requerimientos_LED_010611_e1c3d71a.pdf>

[10] Wikipedia <http://www.wikipedia.org>

ANEXO I

Estudio económico de instalación del sistema de iluminación viaria adaptativa en una instalación existente

Como ya he dicho, este proyecto no es una solución integral para un proyecto de iluminación viaria. Para estimar el coste que supondría instalar este sistema vamos a tomar como ejemplo una calle en la que ya tengamos la instalación de luminarias hecha, como proyecto de mejora de la iluminación en dicha vía.

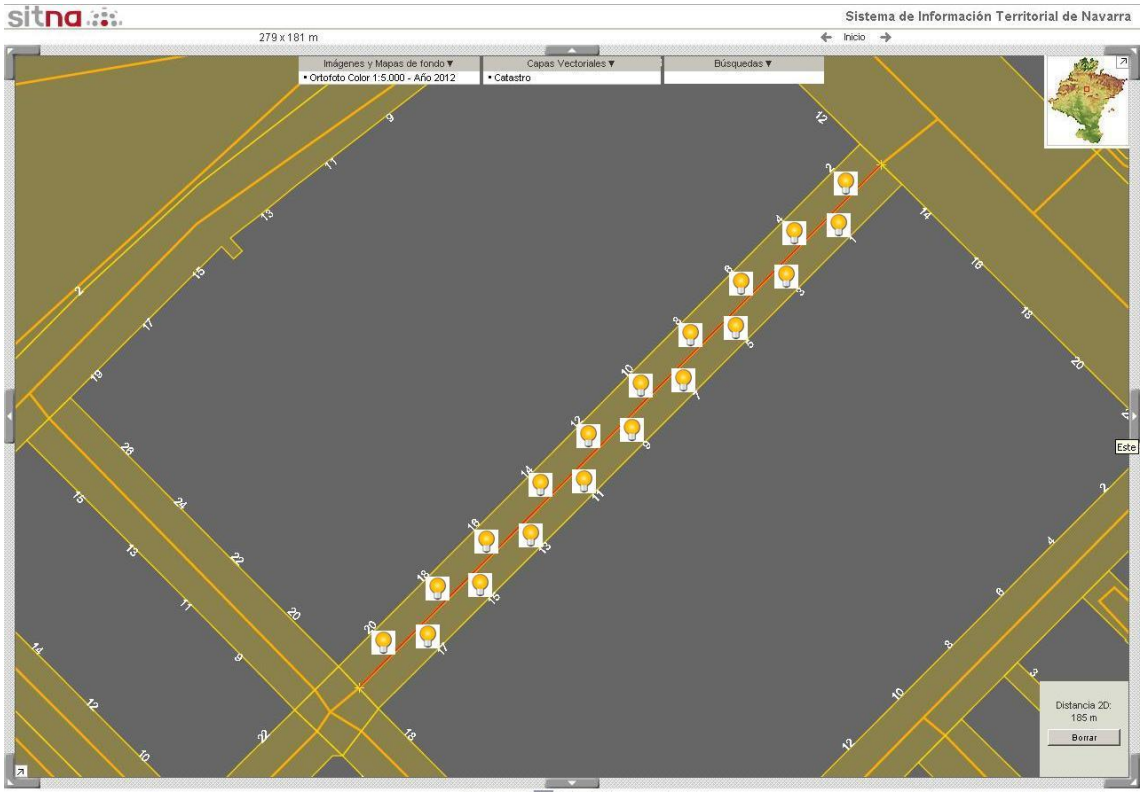


Figura 18: plano de la vía donde se realiza el estudio

Como ejemplo, un tramo de calle de 185m de longitud. Dispone de un total de 19 luminarias dispuestas más o menos sobre cada uno de los portales. Sin entrar en aspectos luminotécnicos, son lámparas de vapor de sodio de alta presión de 250W.

El consumo unitario por año depende de las horas que estén funcionando.

250W X 19 luminarias X 4287h de funcionamiento al año = 20363250W totales consumidos por todas las luminarias durante todo un año.

Se ha de tener en cuenta que el consumo de 250W corresponde a que las luminarias funcionen al 100% de su potencia.

Vamos a estimar que al menos, de media, todas las noches hay 4 horas acumuladas en las que las luminarias pueden estar en reposo, es decir, que no hay tránsito por la calle. En total al año suponen 1460h.

Este tipo de luminarias no pueden, por construcción, estar a una potencia inferior al 60%, por lo que consideraremos esta la posición en reposo y aplicaremos durante todo el tiempo estimado este consumo.

250W X 19 luminarias X (4280-1460)h de funcionamiento al 100% durante un año = 13395000W

+

150W X 19 luminarias X 1460h de funcionamiento al 60% durante un año = 4161000W

Total = 17556000W

La diferencia de W consumidos durante todo un año es 2807250W = 2807,25 kW.

A precio del mercado libre, 0,138658€/kW esto supone un ahorro de 389,24€, siendo el consumo total sin el sistema de 2823,52€ y con él a 2434,27€. Esto supone un 13,79% de reducción en la factura eléctrica.

Para ello, será necesario un equipo completo de 19 motas, compuestas por: placa base, antena, placa de eventos y sensores de presencia. Además una de las placas tendrá instalado un sensor de luminancia.

Los costes de estos dispositivos este año son:

	€/u	**Un.**	**Total**
Placa base	135 €	19	2565 €
Placa de eventos	75 €	19	1425 €
Sensor de presencia	18 €	19	342 €
Luminosidad	2,5 €	1	2,5 €
TOTAL			4334,5 €

Sin tener en cuenta el coste de desarrollo del software ni de instalación ni mantenimiento, la inversión no se amortizaría hasta el 13º año de funcionamiento.

Estos resultados no son nada alentadores para propiciar la implantación de este sistema en instalaciones similares ya existentes, no obstante sería interesante plantear un cambio completo del sistema de iluminación, con otras luminarias que consuman menos, como pueden ser las de tecnología LED y conseguir un ahorro combinado superior.

ANEXO II

Código de las placas

- Placa coordinadora con sensor de luminosidad y presencia

```
uint8_t netAddr[2] ={0x12, 0x34};
uint8_t i;
uint8_t panid[8]={8,7,6,5,4,3,2,1};
uint8_t j;

void setup(){
  //Iniciar USB
   USB.begin();

   //Iniciar el modulo ZIGBEE
   xbeeZB.init(ZIGBEE,FREQ2_4G,NORMAL);

   //Encender el modulo ZIGBEE
   xbeeZB.ON();

   //Activar API
   xbeeZB.setAPI(0x02);

   //(Funcion) Imprime la MAC del dispositivo.
   printMac();

   //Asignar PANID al dispositivo
   xbeeZB.setPAN(panid);

   //Elegir un canal
   xbeeZB.getChannel();

   xbeeZB.setOwnNetAddress(netAddr[0], netAddr[1]);
   xbeeZB.getOwnNetAddress();

   xbeeZB.getOperatingPAN(); // PAN-ID

////////////////////////////////////////

   //Encender la placa de eventos
   SensorEvent.setBoardMode(SENS_ON);
   //Encender reloj
   RTC.ON();
   //configurar la placa
   SensorEvent.setThreshold(SENS_SOCKET3, 1.5);
   XBee.println("Placa de eventos preparada.");

////////////////////////////////////////
```

```
   USB.println("Sistema preparado.");
   delay(5000);

}
packetXBee* paq_sent;
 int8_t state=0;
 long previous=0;
struct packetXBee * packet = NULL;
char * addr = NULL;
char a_addr[5];
char * dirMAC = "12";

long seq = 0;
char message[30];
char respuesta[10];
int respuestaenter;

char *uno="1";
char *dos="2";
char *tres="3";
float luminosity = 0;
int  lum  = 0;
int error;
int intentos=0;
//char temperaturarecibida;
//long temperaturafinal;

////////////////////////////////////////////////////////////////////////////////////////

void loop(){
/////////////////////////////////////////////////////////////
   if (SensorEvent.intFlag & SENS_SOCKET7){
      USB.print("  Presencia detectada. Valor: ");
      float value;
      //Leer el sensor
      value = SensorEvent.readValue(SENS_SOCKET7);
      //USB.print(" Presence value -> ");
      USB.println(value);
      USB.println();
      sprintf(message,"#@3--%s@#\n",dirMAC);
      error = xbeeZB.send_to("0013A200406075B0",message,0x51,NON_BLOCK);
      USB.println();
   }
/////////////////////////////////////////////////////////////

   //Leer la luminosidad
   //Leer en float pero lo pasamos a int
   luminosity = leerLuminosidad();
   lum=luminosity;
   //Copiar (convertir) a un string el valor de luminosidad
   USB.print("Luminosidad: ");
```

```
   USB.println(lum);
   USB.println();
   sprintf(message,"#@1--%d@#\n",lum);
   error = xbeeZB.send_to("0013A200406075B0",message,0x51,NON_BLOCK);
////////////////////////////////////////////////////////////////////
paq_sent=(packetXBee*) calloc(1,sizeof(packetXBee));
paq_sent->mode=UNICAST;
paq_sent->MY_known=0;
paq_sent->packetID=0x52;
paq_sent->opt=0;
xbeeZB.hops=0;
xbeeZB.setOriginParams(paq_sent, "5678", MY_TYPE);
xbeeZB.setDestinationParams(paq_sent, "0013A200406075B0", message,
MAC_TYPE,DATA_ABSOLUTE);
xbeeZB.sendXBee(paq_sent);
if( !xbeeZB.error_TX )
{
XBee.println("sent ok");
}else{
 XBee.println("sent error");
}
free(paq_sent);
paq_sent=NULL;
////////////////////////////////////////////////////////////////////
   USB.println();
   delay(1000);
}
////////////////////////////////////////////////////////////////////////////////////
float leerLuminosidad(){
   float volts = 0;
   float R;
   float luxes;
   //Habilitar interrupciones de la placa de eventos
   SensorEvent.attachInt();
   SensorEvent.detachInt();
   SensorEvent.loadInt();

   //Read the sensor
   volts = SensorEvent.readValue(SENS_SOCKET3);

   //Convertir a LUX
   R=10*(3.31-volts)/volts;
   luxes=pow(10,((log10(R)-1.89)/-0.5));

   return luxes;
}
////////////////////////////////////////////////////////////////////////////////////
void printMac(){
  uint8_t value;
  uint8_t z;

  if(xbeeZB.getOwnMacHigh() == 0 && xbeeZB.getOwnMacLow() == 0)
```

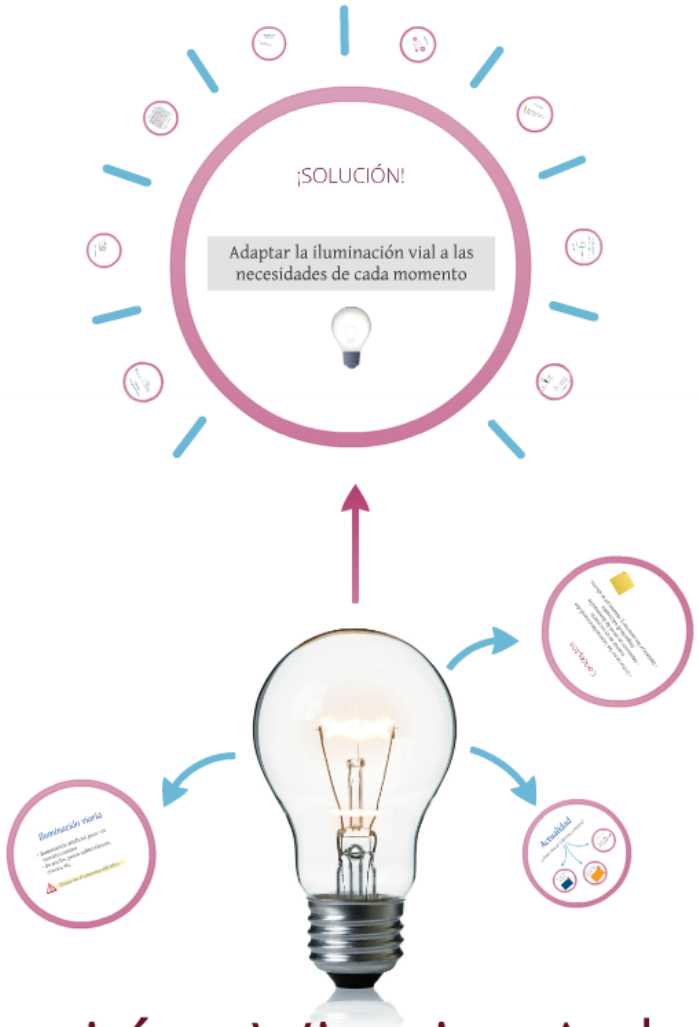
```
 {
  USB.println(" - MAC Address: ");
  for(z=0;z<4;z++)
  {
   value = (uint8_t)xbeeZB.sourceMacHigh[z];

   if(value == 0)
   {
    USB.print("00");
    //sprintf(dirMAC,"%s00",dirMAC);
   }
   else
   {
    USB.print(value,HEX);
    //sprintf(dirMAC,"%s%s",dirMAC,value);
   }
   USB.print(" ");
  }

  for(z=0;z<4;z++)
  {
   value = xbeeZB.sourceMacLow[z];

   if(value == 0)
   {
    USB.print("00");
    //sprintf(macaddr,"%s00",macaddr);
   }
   else
   {
    USB.print(value,HEX);
    //sprintf(macaddr,"%s%s",macaddr,value);
   }

   USB.print(" ");
  }
  USB.println();
  USB.println();
 }
}
```

- Placa EndDevice con sensor de temperatura

```
uint8_t j,k;
uint8_t time[2]={0x03, 0x00};
uint8_t panid[8] = {8,7,6,5,4,3,2,1}; // PAN-ID
char message[30];
int8_t state=0;
long previous=0;

uint8_t coord_addr[2];
uint8_t na_addr[2];

void setup()
{
   ACC.ON(); // inicializamos el ACC

 USB.begin();
 USB.println("Encendiendo el sistema");
 //Inits the XBee 802.15.4 library
 xbeeZB.init(ZIGBEE,FREQ2_4G,NORMAL);


 xbeeZB.ON();

 xbeeZB.setAPI(2);

 xbee802.setPowerLevel(0); // Power Level

 USB.println("Connecting...");

 while(connect() != 0){ // nos conectamos - funcion connect
  delay(500);
 }

 if(xbeeZB.getParentNetworkAddress() == 0)
 {
  coord_addr[0] = xbeeZB.parentNA[0];
  coord_addr[1] = xbeeZB.parentNA[1];
 }
 else
 {
  USB.println("There was a problem while getting the parent Network Address.");
  while(true){}
 }

 if(xbeeZB.getOwnNetAddress() == 0)
 {
  na_addr[0] = xbeeZB.sourceNA[0];
  na_addr[1] = xbeeZB.sourceNA[1];
 }
 else
 {
```

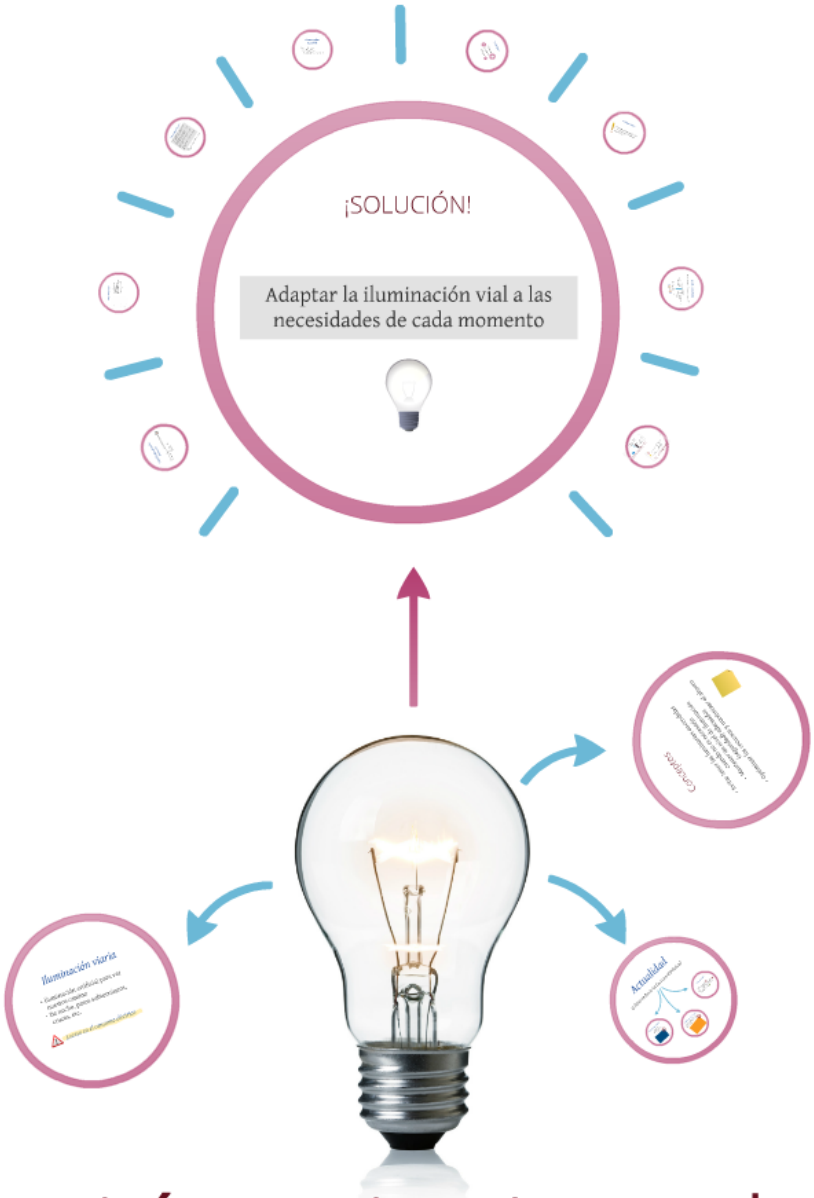
```
    USB.println("There was a problem while getting the Network Address.");
    while(true){}
  }

  Utils.setLED(LED1,LED_ON);

////////////////////////////////////
    USB.println("Inicializando...");
    //Encender la placa de gases
    //SensorGas.setBoardMode(SENS_ON);
    USB.println("Placa de gases preparada.");
////////////////////////////////////

}

  long seq = 0;
  float temp;
  int temperatura;
void loop()
{
 //temp = SensorGas.readValue(SENS_TEMPERATURE);
 temp = RTC.getTemperature();
 //temp = (temp - 0.5) * 100;
 USB.print("Temperatura leida: ");
 USB.println(temp);
 temperatura=temp;
 sprintf(message,"#@2--%d@#\n",temperatura);
 xbeeZB.send_to("0013A200406075B0",message,0x52,NON_BLOCK);
 USB.println();
 delay(2000);
}

uint8_t connect(void)
{
  //xbeeZB.setPAN(panid);
  xbeeZB.setScanningChannels(0xFF,0xFF); // Set List of Channels to scan.
  xbeeZB.setDurationEnergyChannels(6); // Set exponent of scan channel duration
  xbeeZB.getAssociationIndication(); // Check if creating process success

  if(!xbeeZB.associationIndication) // si no nos asociamos bien a la red
  {
    USB.println(" - Node joined to the network!");
    xbeeZB.getChannel();
    USB.print("Channel: ");  // CHANNEL
    USB.println(xbeeZB.channel,HEX);
    USB.println(xbeeZB.channel,DEC);
    xbeeZB.getExtendedPAN();
    USB.print(" - Extended PAN-ID: "); // PAN-ID
    for(j=0;j<8;j++)
    {
      USB.print(xbeeZB.extendedPAN[j],HEX);
    }
```

```
  USB.println();

  xbeeZB.getOperatingPAN();
  USB.print(" - Operating PAN-ID: "); // Operating PAN-ID
  USB.print(xbeeZB.operatingPAN[0],HEX);
  USB.print(xbeeZB.operatingPAN[1],HEX);
  USB.println();

  xbeeZB.getOwnNetAddress(); // Mi direccion de red
  USB.print(" - Net Address: ");
  USB.print(xbeeZB.sourceNA[0],HEX);
  USB.print(xbeeZB.sourceNA[1],HEX);
  USB.println();

  xbeeZB.writeValues(); // escribimos los valores

  return 0x00; // devolvemos un 0 ya que todo ha ido bien
 }
 else // no nos hemos asociado a la red
 {
  USB.print(" - Node not joined to the network (");
  USB.print(xbeeZB.associationIndication,HEX);
  USB.println(")");

  return 0x01; // devolvemos un numero distinto de 0
 }
}
```



Iluminación Viaria Adaptativa

Iñaki Etxeberria Ruesta



Iluminación Viaria Adaptativa

Iñaki Etxeberria Ruesta

Iluminación viaria

- Iluminación artificial para ver nuestro camino
- De noche, pasos subterráneos, cruces, etc.

Exceso en el consumo eléctrico

Actualidad

¿cómo reducir la factura eléctrica?

Programación horaria

Las luminarias se encienden a unas horas fijadas por el orto y ocaso de cada día

Interruptor crepuscular

Cuando la luminosidad baja del umbral prefijado, se encienden las luminarias permanentemente

Otras soluciones

- Doble temporizador ✗
- Apagar un circuito de luminarias ✗
- Sustitución de luminarias por otras de menor consumo ✓

Programación horaria

Las luminarias se
encienden a unas horas
fijadas por el orto y ocaso
de cada día

Interruptor crepuscular

Cuando la luminosidad baja del umbral prefijado, se encienden las luminarias permanentemente

Otras soluciones

- Doble temporizador 
- Apagar un circuito de luminarias 
- Sustitución de luminarias por otras de menor consumo 

Conceptos

- Evitar tener las luminarias encendidas cuando no es necesario
- Mantener un nivel de iluminación (seguridad) adecuados
- Optimizar los recursos y maximizar el ahorro

¡SOLUCIÓN!

Adaptar la iluminación vial a las necesidades de cada momento

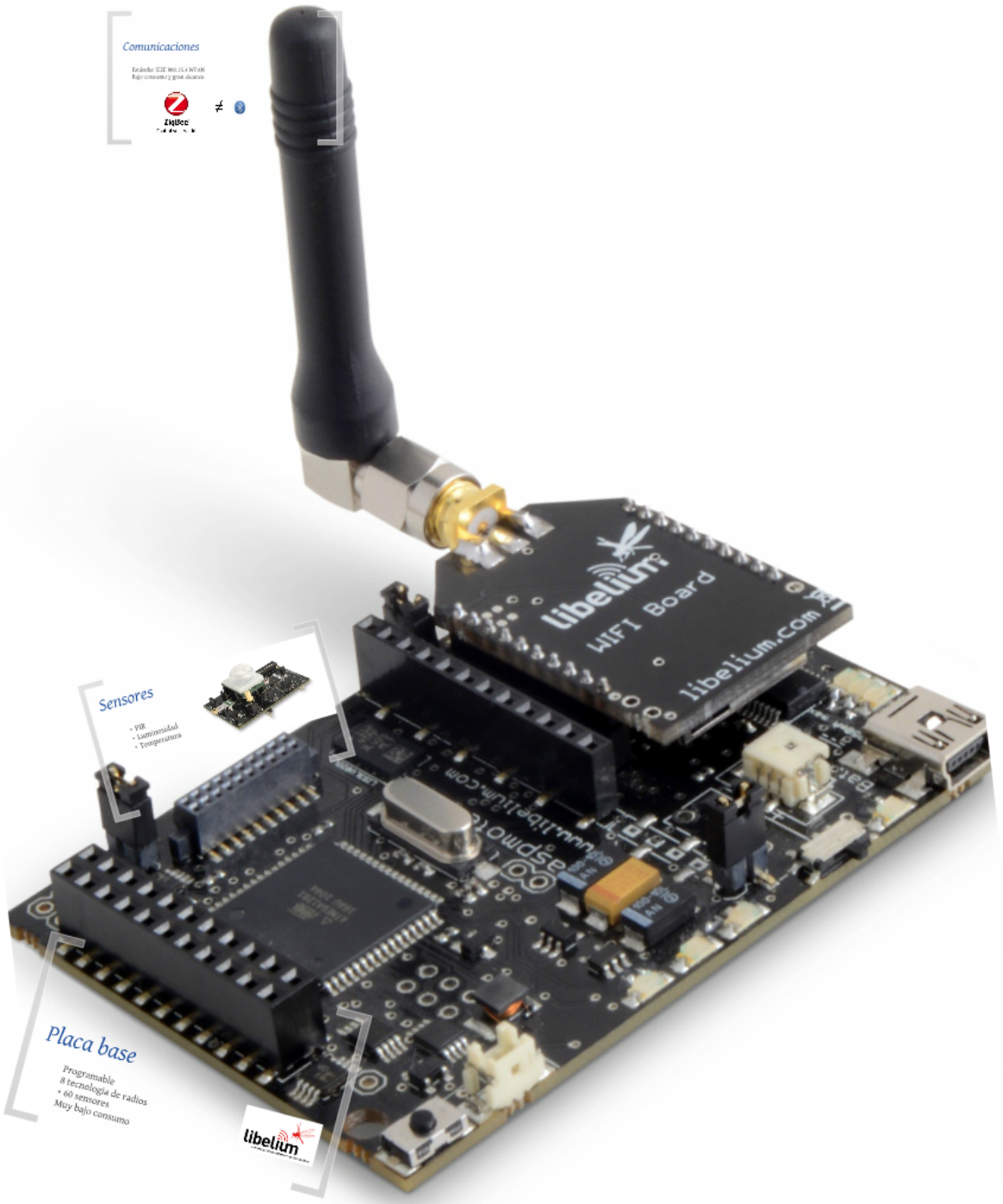
Necesidades de cada momento

Las necesidades varían constantemente y dependen de:

- Luminosidad
- Tráfico

Monitorización

Una red de sensores controlará constantemente la luminosidad y el tráfico y regulará la potencia de cada luminaria



Una r
contr
const
lumir
regul
cada

Placa base

Programable
8 tecnologia de radios
+ 60 sensores
Muy bajo consumo

libelium
comunicaciones inalámbricas distribuidas

Sensores

- PIR
- Luminosidad
- Temperatura

Comunicaciones

Estándar IEEE 802.15.4 WPAN
Bajo consumo y gran alcance

ZigBee®
Control your world

≠

Funcionamiento

1- Cuando la luminosidad baje de un umbral establecido, el sistema se activará

2- En modo reposo, si se detecta presencia, la potencia de la luminaria aumentará

3- Cuando la presencia desaparezca el sistema volverá a reposo

4- Si hay suficiente luminosidad, el sistema se desconecta

Comunicación entre placas constante

Metodología de desarrollo

Programación extrema

- Análisis sencillo
- Revisión constante
- Código simple y auto-comentado

Requisitos

Funcionales

No funcionales

Adicionales

Funcionales

- Sin intervención humana
- Reducir el consumo
- Iluminar la vía adecuándose a las circunstancias: luz y tráfico
- Funcionamiento coordinado

No funcionales

- Fácil mantenimiento
- Rentable
- Código fuente sencillo

Adicionales

Waspmote + Zigbee

Limitaciones

No es una solución integral, no se tienen en cuenta aspectos luminotécnicos

Estudio económico

Tramo de 180m y 19 luminarias SAP 250W

Consumo anual

Sin el sistema	Con el sistema (+ 4334,5 €)
2823,52€	2434,27€
	Ahorro: 13,79%

Amortización: 12 años

Integrar el sistema en una instalación nueva o aprovechar el cambio por luminarias LED y maximizar el ahorro

Conclusiones

- Facilidad en el desarrollo de soluciones tipo
- Gran flexibilidad y potencial

- Gratificante trabajar con algo desconocido y con aplicación real y actual

Líneas futuras

1- Integración PC
2- Sistema distribuido en las motas
3-Otros productos Libelium

Detección de fallos y avisos
Predicción tráfico
Aplicaciones específicas

Conclusiones

- Facilidad en el desarrollo de soluciones tipo
- Gran flexibilidad y potencial

- Gratificante trabajar con algo desconocido y con aplicación real y actual

Problemas surgidos

Sobre todo en la comunicación entre dispositivos

- M2M
- Waspmote - PC

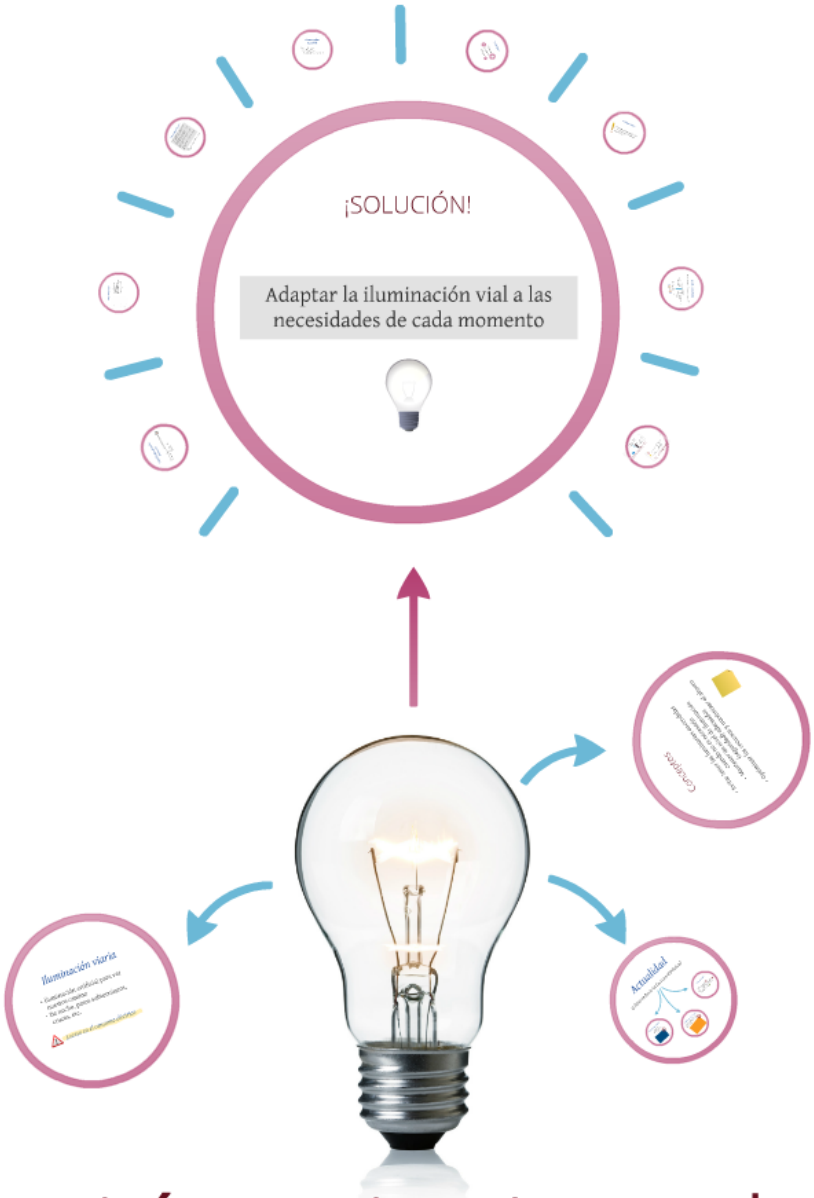
Sensores

- No funcionaban correctamente
- Rango PIR escaso

Líneas futuras

1- Integración PC
2- Sistema distribuido
en las motas
3-Otros productos Libelium

Detección de fallos y avisos
Predicción tráfico
Aplicaciones específicas



Iluminación Viaria Adaptativa

Iñaki Etxeberria Ruesta